UNIVERSITÉ DE MONTRÉAL

CENTRE DE RECHERCHE SUR LES TRANSPORTS

CASE POSTALE 6128, STATION A, MONTRÉAL, QUÉBEC, H3C 3J7

89-01199/

EMME

SYSTEMS PROGRAMMERS GUIDE

VOLUME I

by

Linda James-Lefebvre, Claude Achim,
Caroline Fisk and Jacques Lefebvre

This work was supported by a grant from the Research and Development Centre of Transport Canada

Université de Montréal
Centre de recherche sur les transports - Publication # 124
Janvier 1979

Dépôt légal - 1er trimestre 1979

TABLE OF CONTENTS

VOLUME I

1. Purpose of this guide. 1

2. Organisation of the guide. 1
   - 2.1. Module numbering scheme 4
   - 2.2. File numbering system 5

3. General system description 6
   - 3.1. System structure. 7
   - 3.2. Use of variable dimensions in EMME. 9
   - 3.3. Internal numbering system 10
   - 3.4. Error messages. 13

4. Supervisor 14
   - 4.1. General description of supervisor 14
   - 4.2. Supervisor use of COMMON areas. 15

5. Data bank: Input, validation and scenarios. 17
   - 5.1. General description 17
     - 5.1.1. CORRECTION subcommand. 18
     - 5.1.2. SCENARIO subcommand. 18
     - 5.1.3. USE subcommand 19
     - 5.1.4. PRINT subcommand 19
     - 5.1.5. PUNCH subcommand 19
     - 5.1.6. PURGE subcommand 20
   - 5.2. Data bank structure 21
     - 5.2.1. Data bank index. 23
   - 5.3. Data validation 25
   - 5.4. Data bank module naming scheme. 30
   - 5.5. Data bank use of COMMON areas 32

6. Data bank: Access routines. 33
   - 6.1. General description 33
   - 6.2. Use of COMMON areas 36
   - 6.3. Access routine descriptions 36
     - 6.3.1. Subroutine NAMEOD. 36
     - 6.3.2. Subroutine GTSCAL. 37

7. Bimodal assignment control program 38
   - 7.1. General description of bimodal assignment control program . . 38
     - 7.1.1. Bimodal assignment modules 39
     - 7.1.2. Fixed demand transit assignment modules. 40
     - 7.1.3. Variable demand transit assignment modules 41
     - 7.1.4. Fixed demand road assignment modules 42
     - 7.1.5. Variable demand road traffic assignment modules. 42

8. Transit section. 44
   - 8.1. General description of transit section. 44
     - 8.1.1. Generation of the transit network. 44
     - 8.1.2. Transit path calculations. 45
     - 8.1.3. Generation of transit data for road assignment 47
     - 8.1.4. Iteration control for bimodal assignment 47
     - 8.1.5. Transit assignment 48

9. Road traffic section 52
  9.1. General description of road traffic section 52
    9.1.1. Initialisation of road traffic demand files. 53
    9.1.2. Road traffic assignment. 55
  9.2. Road traffic section: Use of COMMON areas. 57
  9.3. Road traffic section: User specified routines. 59
  9.4. Road traffic section: Vehicle adjustment matrix and upper bound on demand 60

10. Demand function section. 61
  10.1. General description of demand function section 61

11. Graphics programs. 66
  11.1. General description of graphics programs 66
    11.1.1. Vehicle network 69
    11.1.2. Transit network 69
    11.1.3. Vehicle assignment. 69
    11.1.4. Transit assignment. 70
  11.2. Structure of programs. 70
  11.3. Possible user modifications of programs. 70
    11.3.1. CDC specific fortran. 75
    11.3.2. Numerical information 75
    11.3.3. CDC library routines. 75
    11.3.4. CDC plot software 76
    11.3.5. Program limitations 77

12. Report programs. 78
  12.1. General description. 78

13. Segmented (overlay) load 80

14. User supplied routines 86
  14.1. User supplied routine: Function TURLIN. 87
  14.2. User supplied routine: Subroutines LINKIN and FINDLC 88
  14.3. User supplied routine: Function BUSROD. 91

VOLUME II: Appendices

A: COMMON area descriptions. 98

B: Module specifications 153

C: File descriptions 491

D: Cross reference module and file index 563

E: Variable demand traffic assignment algorithm. 586

1. Purpose of this guide

The guide is intended to aid a systems programmer who is responsible for modifying or adding to the EMME system.

The reader is assumed to be familiar with the EMME Users' guide and with the Fortran language and file structures.

Installation of the EMME system is described in separate documentation.

2. Organization of the guide

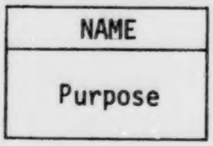
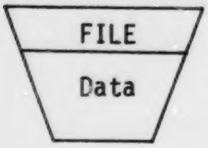
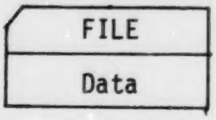
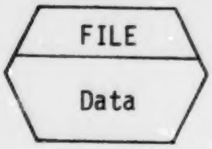
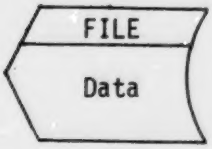
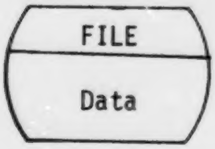
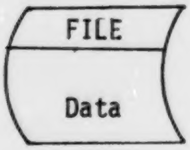
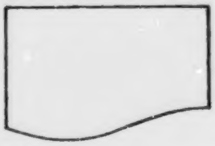
The "General System Description" gives details of the overall EMME system structure and includes details of the variable dimensions used in the EMME system and a description of the internal numbering of the nodes of the two networks (transit and road traffic). The EMME system may be divided into ten sections, namely:

- Supervisor
- Data bank input, validation and scenarios
- Data bank access routines
- Bimodal assignment control
- Transit section
- Road traffic section
- Demand function section
- Graphics programs
- Report programs
- User coded routines

The guide contains a general description for each of these sections together with some general comments on their use of COMMON areas if this is relevant. Details of the COMMON areas can be found in Appendix A, "COMMON area descriptions". The individual modules contained in the sections of the system, are described in Appendix B "Module Specifications" which is organized by module number according to the module numbering scheme described below (2.1.) The files which form the interface between modules/sections are described in Appendix C, "File descriptions" which is organized by file number according to the scheme described in 2.2. The correspondence between module or file names and numbers is contained in Appendix D, "Cross reference module and file index". In some cases several modules are described in one module description. If Appendix B does not contain a separate description for a given module, the module description which includes the given module may be determined from Appendix D.

Flowcharts are included to show the logic for a few of the modules. The standards used for these flowcharts are shown in Figure 2-1.

The transit section and road traffic section contain routines which must be provided by the user. Comments on these routines can be found in sections 8, 9 and 14 of the guide.

Symbol	Block type	Details
	Processing	Program or subroutine block. NAME is program name or main subroutine name
	Input Data File	User specified data which is accessed from the data bank
	Direct Entry Input File	User specified data which is accessed from a non-data bank file
	Intermediate File (Sequential)	Data file created by the system and kept
	Intermediate File (Random)	An Intermediate File accessed using CDC Mass Storage routines or IBM Direct Access routines
	Work File (Sequential)	Data file written and read by a program in the system but not kept after the program execution is complete
	Work File (Random)	A Work File accessed using CDC Mass Storage or IBM Direct Access routines
	Printed report	

Note: FILE is File name

Figure 2-1: EMME Flowcharting standards

2.1. Module numbering scheme

Each module, which may be a main program, subroutine or function subprogram is assigned a number of the form x.yy

where x is the main classification of the module
yy is the number within that classification.

The classifications are:

0	General subroutines
1	Major control program (supervisor)
2	Input and validation of user data
3	Data base access routines
4	Bimodal assignment control program
5	Transit routines
6	Road traffic routines
7	Demand function processing
8	Graphics programs
9	Report programs.

This classification has been used to number all the program modules for convenience only and does not correspond exactly with the sections described above in section 2

The program number appears on any flow chart. for example:

6.11	AUTRIP
Determine total vehicle trips	

The subroutine AUTRIP has the number 6.11 and is one of the road traffic routines.

The program number is also contained in a comments card in the Fortran code of the module for example:

```
C      EMME MODULE      5.01
```

2.2. File numbering system

Each file is assigned a number of the form xss (no period)

where	x	is the main classification of the module which creates the file (see 2.1. module numbering system).
	ss	is the sequence number of the file with the main classification x

Files are further described in the introduction to "File description" in Appendix C.

3. General system description

The EMME system consists of a supervisor which decodes the EMME command/subcommand language and directs control to the relevant section of the system to execute the operations implied by that command. The header cards and data, following a $DATABANK command, through which the user specifies his network characteristics, socio-economic activities and demand function coefficients are processed by the data bank input routines which enter the data into the data bank (see section 5.). An assignment command may call for demand function processing (variable demand or bimodal assignments; see section10.) and will always call the bimodal assignment control module (see section 7.) which determines which assignment modules to call in order to perform the assignment requested. A report command will be decoded by the supervisor which will call the relevant report program module (see section 12.). All requests for data from the data bank are handled by the data bank access routines (see section 6.). These routines are called by the demand function processing modules, assignment modules and report modules.

The data bank is contained on one random access file (one logical unit) but since this file contains many different types of data (socio-economic variables, bus lines, road links, etc.) the system description considers each type of data as being one file with a separate description in Appendix C.

Work files provide the interface between for example, demand function processing and the assignment routines and between the transit and road traffic assignments. The assignment results are saved on intermediate files which are used by the report programs and may be saved by the user for his own requirements (see EMME Users' Guide).

It should be noted that the graphics program (see Section 11.) are not currently integrated into the EMME system but they do use the standard data bank access routines and use some EMME files. The programs have been written in such a way that they could be integrated into the system for an installation which had graphics capabilities.

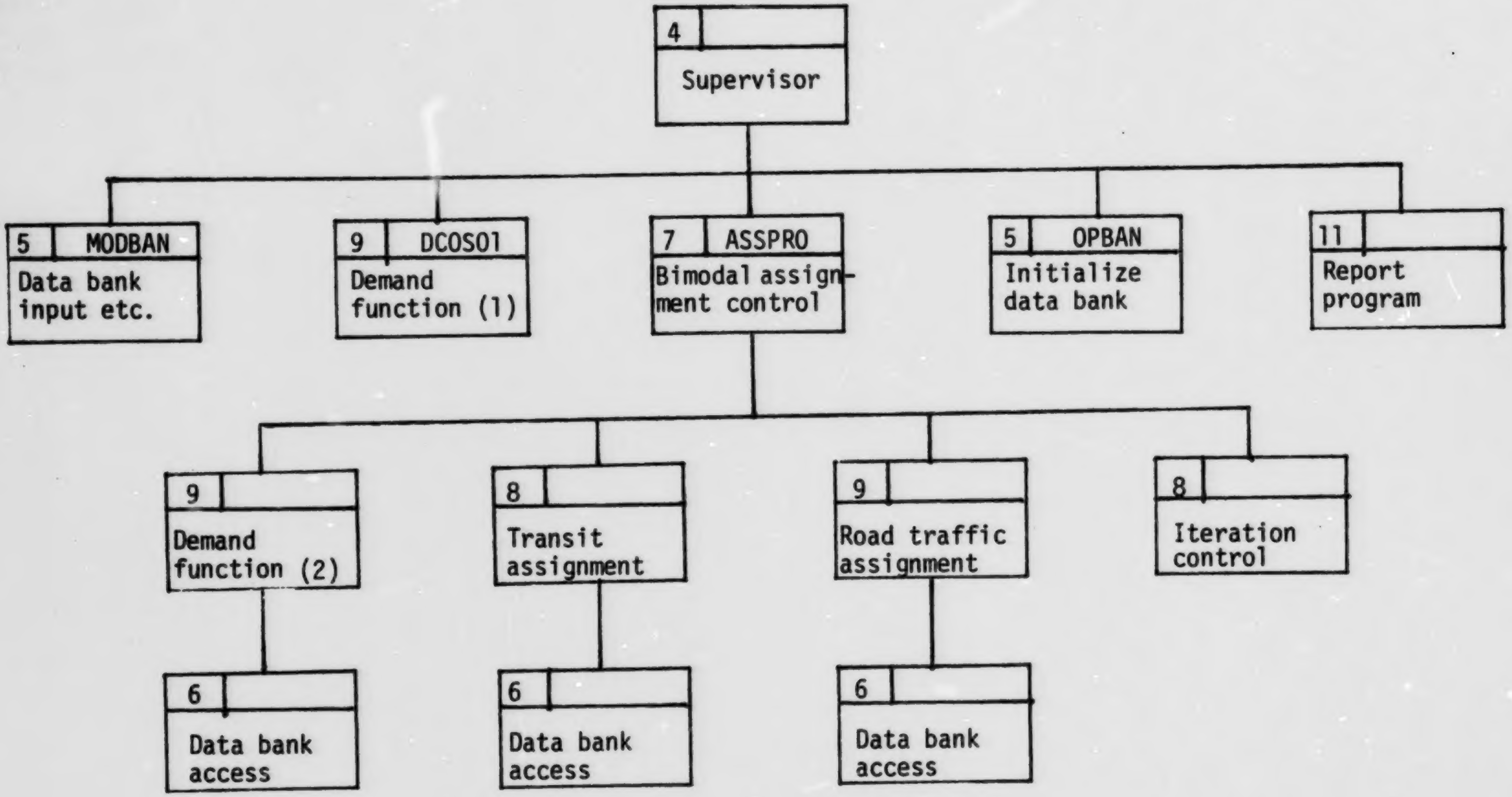
3.1. System structure

The system structure is hierarchical as shown in Figure 3-1 where each block represents a segment of the system containing one or more modules. Thus the supervisor (section 4) may call the bimodal assignment control module (section 7, module 4.00 ASSPRO) which in turn may call the demand function processing segment (section 10) and/or the iteration control segment (section 8) and/or the transit assignment segment (section 8) and/or the road traffic assignment segment (section 9). The segments called will depend on the type of assignment (fixed or variable demand or bimodal)

In Figure 3-1 a block which does not indicate a module name represents several modules which may be called. Details may be found in the sections describing the calling and called segment and in the module descriptions, for example 4.00 ASSPRO.

Note that the section numbers refer to sections in the guide and do not correspond with the module numbering system.

The structure shown in Figure 3-1 can also be used to construct overlays in order to execute the system when core storage is limited (see Section 13).


4
Supervisor
5
MODBAN
Data bank input etc.
9
DCOSO1
Demand function (1)
7
ASSPRO
Bimodal assignment control
5
OPBAN
Initialize data bank
11
Report program
9
Demand function (2)
8
Transit assignment
9
Road traffic assignment
8
Iteration control
6
Data bank access
6
Data bank access
6
Data bank access

Numbers indicate sections of the guide which describe this segment.
Names indicate name of module called by supervisor to execute this segment.

Figure 3-1. EMME system hierarchy

3.2. Use of variable dimensions in EMME

The EMME system is coded to allow processing of networks of any size within the limits of the EMME input data format and of the core storage of the machine on which the system is executed. This is achieved by coding the dimensions of arrays with a symbolic name preceded and followed by percentage signs, for example %NODE%. Before the EMME programs are compiled the names are replaced by numbers suitable for the system required by a user. These numbers are then the maximum number of nodes, links, transit lines, etc. which the users' system can handle. The symbolic names are also used to define constants required in the system, for example

```
DATA MAXNOD/%NODE%/
```

initializes MAXNOD to the maximum number of nodes allowed in the users' system.

A complete list of the symbolic names and their significance can be found in Appendix A.

3.3. Internal numbering system.

The user defines a set of (external) node numbers in 3 sections of his input data

- the list of centroids
- the road link definitions
- the transit line definitions.

The internal node numbers are assigned at the following times:

- when all input data or modifications to data in the data bank have been processed without encountering an error (i.e. a command other than $DATABANK is encountered in the input stream)
- when a scenario is "used" (USE scenario) and all additions, deletions or modifications defined by the scenario(s) have been (temporarily) applied to the basic data bank.
- when a "used" scenario is removed (i.e. $END, without the KEEP option, is encountered and a scenario was in use).

The internal numbering is performed by the data bank routines using the sequence 1 to the total number of nodes found in the 3 sections mentioned above. The nodes are ordered as shown in Figure 3-2, that is

- centroids
- turning nodes (road network)
- regular road nodes (road network)
- transit-only nodes (transit network only)

A turning node is defined as a road node found at one end of a turning link in the road network. A turning link is a road link which has a link type considered to be a turning link as determined in subroutine TURLIN (user coded subroutine).

In Figure 3-2 the names NCENT, NTURN, NROAD, etc.. are the names of variables in the COMMON area /NETSIZ/ which contain the values indicated. For example. NCENT contains the number of centroids and is also the highest internal centroid node number since centroids are numbered from 1 to NCENT. NCENT+1 indicates that the number is the value in NCENT plus one.

Figure 3-2 also shows the node numbers used in the different parts of the EMME system. TRAFFIC node numbers are the numbers used by the road traffic assignment modules and are identical to the internal node numbers of the data bank. TRANSIT node numbers are the numbers used by most of the transit assignment modules. The transit modules which use corresponding road links and transit only links use the numbering identified as CORNET.

Number of nodes in section	Data Base node numbers	TRAFFIC node numbers	TRANSIT node numbers	CORNET node numbers
NCENT centroid nodes	1 . . . NCENT	1 . . . NCENT	1 . . . NCENT	unused
NTURN turning nodes (road)	NCENT + 1 . . . . NSPENØ	NCENT + 1 . . . . NNØDE	unused	NCENT + 1 . . . . NNØDES
NRØAD regular road nodes	NSPENØ + 1 . . . . NNØDE		1 . . . . NNØDET	
NTRAN transit-only nodes	NNØDE + 1 . . . NNØDES	unused		

Figure 3-2 - Internal node numbers

3.4. Error messages

Messages produced by the system have the following format

xyynni Text

where xyy = the program module number as described in 2.1 Module numbering scheme

nn = the number of the message within the module

i = I for Informative messages (no error)
= E for Error messages

Text = message text.

Each message is printed starting in print position 1. There is 1 space between i and the text.

4. Supervisor

4.1. General description of supervisor

The supervisor initializes the EMME system (module 1.10 INISUP) and reads and decodes EMME commands and subcommands with the exception of $DATABANK subcommands. It calls the relevant processing routines to execute the decoded commands/subcommands.

A command is read (by module 1.11 RDCOM) in module 1.00 EMSUP or, following a sequence of subcommands, in the module processing the subcommands for example 1.60 DECASS. All commands are decoded in module DECCOM (called from 1.00 EMSUP) which returns the command number in NCOM. If the command allows options or subcommands,EMSUP calls an appropriate routine to decode them and set variables in COMMON areas as described in the decoding modules (1.15 DECDEB, 1.16 DECEND, etc., see module 1.00 EMSUP for decoding module names). For commands which require further processing, EMSUP calls the processing modules. For example, for $ASSIGN,EMSUP calls 7.01 DCOSO1 (variable demand and bimodal assignments only) and then 4.00 ASSPRO, the bimodal assignment control module. Details for other commands can be found in module 1.00 EMSUP.

The command $REPORT is handled differently from other commands that require further processing. In the same way as other commands which allow subcommands, module DECREP is called to decode the subcommands. However, in addition subroutine DECREP (as opposed to EMSUP) also calls the routine to process the subcommands (6.94 PRTLNK, 6.92 PRTGPQ, 6.96 RTRIPD). When DECREP encounters a command it returns to EMSUP to decode that command.

For commands other than $REPORT,control is returned from the processing modules to EMSUP to decode the next command.

Any errors which may require the EMME system to be terminated, or to modify subsequent processing are indicated via the parameter OK. The logical variable OK is initialized to TRUE by 1.10 INISUP and may be set to FALSE by any of the decoding or processing modules which find a "catastrophic" error. An error message is printed by the decoding or processing module. Conditions which cause EMME to be terminated are:

- databank unusable
- more than 10 command errors.

Errors in decoding cause processing to be suppressed but the supervisor continues to read and decode commands and subcommands.

All the decoding routines are table driven which allows for easy expansion of the lists of command, subcommand, option and parameter codes. The format of these tables is contained in the description of module 1.12 ANWORD in Appendix B.

4.2. Supervisor use of COMMON areas

COMMON area	Use
/APARM/	Filled with decoded $ASSIGN options and parameters for subsequent use by the assignment routines
/CARTE/	Filled by 1.11 RDCOM and by databank input routines. Contains the current command or subcommand being decoded. Print indicators prevent reprinting of the same information. An end-of-file indicator indicates that EOF has beed reached on the command input file (COMFIL).

/CONST/	Contains constants. Initialized by the supervisor for use by the databank and assignment routines.
/REP/	Filled with date, time and also title text decoded from $TITLE.
/SUPCOM/	Supervisor control switches. Also OD matrix number decoded from a $ASSIGN parameter.
/UNIT/	Logical unit numbers of all files,initialized by supervisor.

5. DATA BANK: INPUT, VALIDATION AND SCENARIOS

5.1. General description

The data bank input and scenario handling routines are called from modules 1.02 MODBAN and 2.120 REMS. Both modules are called from 1.00 EMSUP.

REMS calls the routines required to remove a scenario from the basic data bank file. The USE scenario subcommand is processed by routines called from MODBAN.

MODBAN calls the routines which handle the subcommands, headers and data which follow a $DATABANK command in order to create the logical files in the databank from the users data. The structure of the databank is described in paragraph 5.2. An outline of the function of MODBAN follows.

MODBAN checks for the option NEW on the $DATABANK command. For a NEW data bank the data bank index, etc. are initialized, otherwise the data bank is assumed to exist and the index is read from the first record on file BANK.

MODBAN then reads and decodes the subcommand following the $DATABANK command. Processing of any subsequent headers and data depends on the subcommand as described in the following paragraphs. The validation performed for the data following CORRECTION and SCENARIO subcommands is described in paragraph 5.3.

When all the subcommands, headers and data have been processed and if the data bank has been modified, MODBAN calls module 2.119 IJ to sort and renumber and validate sections of data in the data bank depending on the modifications already made, if any, as indicated in vector BETA in COMMON /BG/ (see file 336 DBSTAT in Appendix C) and to update the data bank index.

The node tables (file NODES; module 2.60 KEEPND) are rewritten if they have been modified. The scheme for naming the modules in the data bank section of the EMME system is described in paragraph 5.4.

5.1.1. CORRECTION subcommand

MODBAN reads and decodes each header (module 2.09 HEADER) and calls the appropriate routine to read and process the data following the header (see Figure 1.02-1 in description of 1.02 MODBAN in Appendix B). The routine reads and validates each data record and, if the data is valid, modifies the data bank. The modification code in column 1, A for add, M for modify, D for delete determines the type of validation performed and, if the data is valid, the nature of the 'correction' in the data bank. Each routine sets a flag in the vector BETA in COMMON /BG/ to indicate the status of the current attempt at modification, see description of file 336 DBSTAT. The routine returns control to MODBAN when an END card is encountered. MODBAN then reads and decodes another header and the process is repeated until a $DATABANK subcommand or an EMME command is encountered.

5.1.2. SCENARIO subcommand

MODBAN picks up the name from the SCENARIO subcommand and then calls 2.98 SCENO to store the name in the data bank and initialize the accumulation of the scenario records which follow the subcommand. The scenario headers and data are read and validated using the same routines as for a CORRECTION subcommand, but the modifications indicated in the data are not applied. The routines instead call 2.99 ACCUM to accumulate the records into the file SCEN. The last records are accumulated using routine 3.54 SCENF called from MODBAN.

5.1.3. USE subcommand

MODBAN picks up the name of the scenario from the SCENARIO subcommand and calls 3.41 SCEN1 to find the scenario in the data bank. MODBAN then gets a header from the scenario (module 3.40 MUCCA) decodes it and calls the appropriate routine to read the data following the header from the scenario and process it (see Figure 1.02-1 in description of 1.02 MODBAN in Appendix B). The routine reads each data record from the scenario and modifies the data bank in the same way as for a CORRECTION subcommand except that the data bank is temporarily modified. The modifications are performed by the same routines as for a CORRECTION subcommand. The routine returns control to MODBAN when an END card is encountered in the scenario. MODBAN then reads another header from the scenario and the process is repeated until the end of the scenario is reached. A header may be INCLUDE which causes the included scenario to be put in the scenario stack by module 3.41 SCEN1 and initialized for reading by module 3.42 SCEN2 after which processing of scenario header and data continues as above. At the end of the included scenario module SCEN2 is called to restart processing of the scenario which contained the INCLUDE. When the scenario stack is empty MODBAN reads another subcommand as described in 5.1.

5.1.4. PRINT subcommand

MODBAN calls 3.56 PRINT to decode the header which is part of the subcommand and to print the data corresponding to the header.

5.1.5. PUNCH subcommand

MODBAN calls 3.75 PUNCH to decode the header which is part of the subcommand and to punch the data corresponding to the header.

5.1.6. PURGE subcommand

MODBAN calls 3.26 PURGE to decode the header which is part of the subcommand and to purge the data corresponding to the header. PURGE calls module 2.62 B to clean up parts of the data bank affected by the purge.

5.2. Data bank structure

The data bank is a collection of logical files used by the assignment and report programs. Each of these logical files contains records as described in the file descriptions numbered between 200 and 399 inclusive. However the data bank can be considered as a file in itself and, in fact, it is a single file from the point of view of the computer operating system.

Thus the collection of logical files contained in the data bank are merely records of the latter. The structure of these "records" is extremely variable; it depends on which file is considered.

In order to achieve some understanding of the data bank, the following structure has been designed.

Any information in the data bank is part of a block. A block is an area of fixed length read or written in a single operation. This length has been set to 511; it could be changed* if the data differed greatly from that of the city of Winnipeg. The length of a block is represented by the symbol %L.BLOC%.

In all cases, the first word of a block contains the position of the last useful word of information within the block. Thus information can be added without destruction. The minimum value contained in the first word is 1, which means that the block contains no information except its own length.

The last word of any block is a pointer to another block containing subsequent information. The last block of a chain contains 0 in position %L.BLOC%.

---

* See section 3.2 and Appendix A.

Words 2 to %L.BLOC%-1 are available for data.

When a block is canceled by means of a PURGE command or in any other way, it is linked with another empty re-usable block. This is intended to keep the volume of the data bank to a minimum. In this case, the first work of the block contains the number of another free block and the last one in the chain contains 0.

"Number of a block" means the number used by the operating system to retrieve the block. On the CDC computer, this is the PRU (Physical Record Unit). On the IBM computer, it is the record number in the sense of the DEFINE FILE statement.

There is an exception. The first block of the data bank is an index. Its length is 63 words. It contains some information relative to the status of the data bank and the number of the first block of each logical file. The index allows access to the beginning of a file and from there the following blocks can be retrieved. The content of the index is detailed in paragraph 5.2.1.

In summary, the blocks are the physical records of the data bank. A chain of blocks is a logical file created from user data or by the system and used by the assignment or report programs.

5.2.1. Data bank index.

The numbers indicate the sequence number of the work in the index.

1 - "CRT"
2 - block number, centroids (file 302 CENT)
3 - number of active centroids
4 - block number, geocoded nodes (raw) (file 304 NODEIN)
5 - block number, geocoded nodes (expanded) (file 200 NODES)
6 - block number, normal road links (file 211 RLINK)
7 - number of active normal road links
8 - block number, access road links (file 211 RLINK)
9 - number of active access road links
10 - block number, egress road links (file 211 RLINK)
11 - number of active egress road links
12 - block number, transit lines (file 201 LINES)
13 - number of active transit lines
14 - block number, transit only links (file 204 TLTIM)
15 - number of active transit only links
16 - block number, access transit links (file 203 ACCESS)
17 - number of active access transit links
18 - block number, egress transit links (file 203 EGRESS)
19 - number of active egress transit links
20 - block number, correspondence table (file 202 NODCOR)
21 - number of entries in correspondence table
22 - name of O-D matrix # 1
23 - name of O-D matrix # 2
24 - name of O-D matrix # 3
25 -
26 -
27 -
28 - block number, O-D matrix # 1
29 - block number, O-D matrix # 2 (file 221 PERSPQ)
30 - block number, O-D matrix # 3
31 - block number, auto occupancy (file 222 AUTOCC)
32 - block number, modal split (file 223 MODEST)

33 - highest block number used
34 - block number of a re-usable block
35 - indicates if a scenario is applied to basic network
36 - block number, vectors beta and gamma (overall status)(file 336 DBSTAT)
37 - block number of the list of nodes encountered on transit lines (file 337 STOPS)
38 -
39 - block number, road links (access, egress and normal)(file 339 RLNKIN)
40 - number of active road links
41 -
42 - permanent scale
43 - temporary scale
44 - flag of sorted centroids
45 - flag of sorted normal transit links
46 - flag of sorted access transit links
47 - flag of sorted egress transit links
48 -
49 -
50 -
51 -
52 -
53 - status of data bank (working condition or not)
54 - block number, equivalence internal-external node numbers (file 354 NODEXT)
55 - block number, demand function travel time coefficients (raw)(file 355 DCOKIN)
56 - block number, demand function travel time coefficients (expanded) (file 231 DCOK)
57 - block number, zone grouping (file 357 ZGROUP)
58 - number of the last used block, zone grouping
59 - block number, socio-economic variables and coefficients (file 359 SOCIO)
60 - block number, vehicle adjustment matrix (file 224 VEHADJ)
61 - block number, scenarios (file 361 SCEN)
62 - block number, transit path multipliers (raw) (file 362 PMUWSW)
63 - block number, transit path multipliers (expanded) (file 250 PMUW)

5.3. Data validation

The data which follow a CORRECTION or SCENARIO subcommand are validated record by record as the data are read in. The validation depends on the type of data modification (Add, Modify or Delete) specified in column 1 of the record, and on the type of data as specified on the preceding header. A summary of this validation by data type and modification type follows.

Header: CENTR0IDS

Case A

CENTROID: must be integer or absent,
The centroid must not exist yet.
If X present, must be integer, absent is undefined (-1)
Same for Y.
NAME OF ZONE: anything.

Case M

The centroid must exist.
Blank fields are left unchanged; dashes in field cancel the value; in particular, dashes in NAME OF ZONE change the name to blank.
If X, Y and NAME OF ZONE are absent: no effect.

Case D

The centroid must exist.
X, Y and NAME OF ZONE are ignored.

Header: NODE COORDINATES

Case A

NODE must be integer or absent.
The node must not be defined yet.
If X of Y is present, it must be integer.
If X or Y is absent, it is undefined (-1).
If both are absent: no effect.

Case M

The node must be defined.
If X or Y absent, it is left unchanged.
Dashes cancel the value.
If both X and Y absent: no effect.

Case D

The node must be defined.
X and Y are ignored.

Header: ROAD LINKS

Case A

ORIGIN and DESTN must be integer. The link must not exist yet.
We shall know from the list of centroids if it is an access link.
TYPE is integer (absent = 0)
LENGTH is numeric F5.2 (absent = -1)
OBS. TIME is numeric F5.2 (absent = -1)
OBS. VOLUME is numeric F5.0 (absent = -1)
DATA i is numeric F5.0 (absent = 0)
":" announces the beginning of text, not necessarily in column 56.
This would be the case if the card image is entered via a teletype without tabulator.
TEXT: anything.
Symmetric indicator must be "*" or blank. The purpose of * in this field is to automatically generate another card with the same information but the link is then oriented from DESTN to ORIGIN. In this latter case, the reversed link must not exist yet.

Case M

ORIGIN and DESTN must be integer and the link must exist. If * present, modification is done both ways; and both links must exist.
Syntax of other fields as in case A. Dashes cancel the value, blanks leave it unchanged.

Cancellation of LENGTH, OBS.TIME and OBS.VOLUME is -1.
Cancellation of TYPE and DATA i is 0.
Cancellation of TEXT is spaces.

Case D

ORIGIN and DESTN must be present. The link must exist. If * present, both ways are deleted. Other fields are ignored.

Header: TRANSIT LINES

Case A

LINE Nº	must be integer. The bus line must not exist yet.	
NAME	cannot be blank; anything else is good.	
HEADWAY	must be present; it is numeric F4.0.	
LENGTH	is numeric F5.2	(absent = undefined = -1).
SPEED	is numeric F5.2	(absent = undefined = -1).
TYPE	is integer	(absent = defined = 0).
LAYOVER	is integer	(absent = undefined = -1).
VEHICLES	is integer	(absent = undefined = -1).

Following cards must be in sequence.
LINE Nº is repeated on each card; it must be the same as on first card.
NØDE is integer or absent.
DIR must be "A", "R" or blank.

Case M

First card: redefinition of characteristics
LINE Nº must be present. The line must exist. Other fields are left unchanged, are cancelled or replaced depending on blanks, dashes or new values. It is impossible to cancel NAME and HEADWAY. NAME cannot be blank.
Following cards: redefinition of route.
If route is changed, the whole set of cards must be replaced and the first card must be present also. Otherwise only the first card is needed.

Case D

LINE N$^{\underline{o}}$ and NAME must be present. The line must exist. Other fields are ignored.
Complete line description is deleted.

Header: NODE EXPANSION

Case A

TRANSIT NODE must be integer.
CORR ROAD NODES must be integer or absent.
No repetition is allowed. If there is a 0, it must be alone.

Case D

TRANSIT NODE must be integer and defined.

Case M

Validation as described for D and as described for A.

Header: TRANSIT LINKS

Case A

ORIGIN and DESTN must be integer.
The link must not exist yet.
Neither ORIGIN nor DESTN may be a centroid.
Symmetric indicator(S) may be "*" or blank. If *, the link is created in both directions.
TIME and REVERSE TIME are numeric F5.2 or absent.
If S is not blank then REVERSE TIME will be set to the same value as TIME only if REVERSE TIME is absent.

Case M

ORIGIN and DESTN are integer. The link must exist, or both links, if * present.
TIME and REVERSE TIME will have same value if one is absent and * is present.
Dashes cancel the value, but this does not make any sense.

Case D

The link must exist.

If S is * both links must exist and are deleted.

TIME and REVERSE TIME are ignored.

Header: TRANSIT ACCESS LINKS

Same as TRANSIT LINKS except that ORIGIN is a 'zone centroid' and must be a centroid, and DESTN is an 'access node' which must not be a centroid.

Header: OD MATRIX
AUTO OCCUPANCY
MODAL SPLIT
VEHICLE ADJUSTMENT MATRIX

Case A

ORIGIN must be a centroid.

Each DESTN must be a centroid also. The O-D pair must not be defined yet.

TRIPS is numeric F5.0; if absent it is assumed to be 0.

Case M

TRIPS contains the new value. Dashes and 0 produce same effect.

Case D

ORIGIN must be a centroid

If DESTN is absent, the entire line of the matrix disappears.

If DESTN is present the O-D pair is set to 0.

TRIPS ignored.

5.4. Data bank module naming scheme

The modules consist of a prefix followed by a suffix. The prefix is one letter except for the data bank access routines described in section 6 which have a 2 letter prefix. Thus for example the routine which processes a data record to add a centroid into the data bank is called ACEN. 'A' is the prefix for add routines and 'CEN' is the suffix for the routines which process centroid data. Figure 5-1 gives the meaning of the prefixes and suffixes.

Prefix	Meaning
A	Add
B	Clear
D	Delete
I	Renumber i.e. assign internal numbers, sort
M	Modify
O	Read data from a scenario stored in the data bank
P	Print
Q	Query
R	Read user supplied data
S	Same as prefix I
U	Punch
RD	Read data from a data bank file (access routine)
GT	Get data from data bank (access routine)
PS	Position data bank file (access routine)

Suffix	Meaning
CEN	Centroids
DFTTC	Demand function travel time coefficients
GEO	Geocoding (coordinates)
LINE	Transit lines
LNK	Transit links
MAT	Matrices
MINI	Turning nodes, turning links
OD	O-D matrices
PMUW	Transit path multipliers
RL	Road links
SCEN	Scenario
SE	Socio-economic
TL	Transit links
TPM	Transit path multipliers
ZONEG	Zone group

Figure 5-1: Data bank module name prefixes and suffixes

5.5. Data bank use of COMMON areas

The data bank section uses COMMON areas as follows:

COMMON area	Use
MEM	Allocated dynamically to contain input/output buffers, lists, stacks, etc. Managed by module 0.01 GMEM.
COMBAN	Contains the data bank index (see paragraph 5.2.1.). Also contains the pointers to areas in MEM for each logical file in the data bank. Also, for each file, COMBAN contains the block numbers of the first and current block and an end of file indicator (all files except transit path multipliers).
FETBAN	Contains the data bank file table on the CDC system. On both CDC and IBM systems, FETBAN is used to keep the overall status of the data bank and the record number variable.
COMEXT	Contain pointers to vectors in MEM which contain external node numbers.
CONST	Defines the values used for infinity.
PRGFLO	Indicates if data bank is usable. Set during validation.
CARTE	Contains the last record read from file COMFIL and indications of whether it should be printed and if it has been printed. Also contains COMFIL end of file indicator.
RLCOM	Contains counters used to handle road links in the input routines and when assigning the internal node numbers to road links.
ABC	Contains the 26 letters A through Z, comma and equals sign.
COMTPM	Contains the same pointers as COMBAN but applicable to transit path multipliers.
BG	Contains the vectors saved on file DBSTAT. Used to determine if sections of the data bank contain errors or if internal numbers must be reassigned as a result of correcting data or removing scenario(s).
CHRSET	Contains numeric characters 0 through 9.

6. DATA BANK: ACCESS ROUTINES

6.1. General description

The databank access routines are those used by the assignment and report programs to retrieve the data from the logical files in the data bank. In addition the module 3.21 OPBAN 'opens' the data bank, that is it reads the data bank index and determines if the data bank is usable and sets up the external node number vectors and the current network dimensions for use by the assignment and report programs.

Most of the logical files are read sequentially. In this case a routine PS.... positions the file to read the first record (rewind) and records are read by successive calls to a routine RD.... . Random access is provided to some files via routines GT.... which get the data from a specific record defined by one of the parameters of the subroutine call. Following is a list, in file number order, of the routines used to access the logical files. The module number gives the module containing the routine, usually one routine per module. The file number gives the file description containing details of the routine arguments. If the file number is blank, details of the arguments can be found in paragraph 6.3.

Access routine	Module	File	Purpose
GTNODE	3.14	200 +354	Get external node number, and coordinates if any,for a given internal node number
PSLINE	3.77	201	Rewind transit line file LINES
RDLINE	3.01	201	Read data for next transit line
GTLINE	3.02	201	Get data for a given transit line
PSCOR	3.78	202	Rewind correspondence table NODCOR
RDCOR	3.03	202	Read next entry in correspondence table
PSACCL	3.05	203	Rewind transit access link file ACCESS
RDACCL	3.05	203	Read next transit access link
PSEGRL	3.05	203	Rewind transit egress link file EGRESS
RDEGRL	3.05	203	Read next transit egress link
PSTLT	3.05	204	Rewind transit link file TLTIM
RDTLT	3.05	204	Read next transit link
PSLINK	3.15	211	Rewind section of road link file RLINK
RDLINK	3.16	211	Read next road link, standard data
RLINKU	3.18	211	Read next road link, user defined data
PSOD	3.13	221	Rewind OD matrix file PERSPQ
RDOD	3.12	221	Read OD matrix data for next origin
PSOCC	3.13	222	Rewind auto occupancy file AUTOCC
RDOCC	3.12	222	Read auto occupancy data for next origin
PSMEST	3.13	223	Rewind modal split estimate file MODEST
RDMEST	3.12	227	Read modal split estimate for next origin
PSODEX	3.13	224	Rewind vehicle adjustment matrix file VEHADJ
RDODEX	3.12	224	Read vehicle adjustment data for next origin
PSDCOK	3.25	231	Rewind demand function coefficient file DCOK
RDDCOK	3.24	231	Read demand function coefficients for next origin
PSDCOC	3.36	233	Rewind socionomic coefficient file DCOC
RDDCOC	3.23	237	Read socionomic coefficients for next zone
PSDVAR	3.36	234	Rewind socionomic variable file DVAR
RDDVAR	3.23	234	Read socionomic variables for next zone
PSPMUW	3.61	250	Rewind transit path multipliers file PMUW
RDPMUW	3.60	250	Read transit path multipliers for next origin
GTNODE	3.14	354	See GTNODE - file 200

Access routine	Module	File	Purpose
GTNODV	3.33	354	Get vector of external node number
PTNODE	3.20	354	Get internal node number for a given external node number
NAMEOD	3.78	-	Get names of OD matrices
GTSCAL	3.58	-	Get coordinate scale

6.2. Data bank access routines: Use of COMMON areas

The data bank access routines use COMMON areas as follows:

COMMON area	Use
MEM	Allocated dynamically to contain buffers for the logical files and node number lists. Managed by module 0.01 GMEM
COMBAN	Contains the data bank index (see paragraph 5.2.1.). Also contains the pointers to areas in MEM for each logical file, the first and current block number and end of file indicator for each file. (All files except transit path multipliers).
COMEXT	Contains pointers to vectors in MEM which contain external node numbers.
SUPCOM	Transmits the OD matrix number, decoded by the supervisor from a subcommand, to the PSOD routine.
COMTPM	Same as COMBAN but applicable to transit path multiplier files.
BG	Used by OPBAN to determine if any sections of data bank contain errors making it unusable.

6.3. Access routine descriptions

The majority of the data bank access routines are described in the file descriptions of the data they access, as indicated in the list in 6.1. The routines NAMEOD and GTSCAL do not access logical files but parts of the data bank index. Details of their arguments follow.

6.3.1. Subroutine NAMEOD

SUBROUTINE NAMEOD (ODID)

NAMEOD gets the names of the 3 OD matrices as they were entered on the OD MATRIX header. The argument is:

INTEGER ODID(3). ODID(N) will contain the 4 character name of OD matrix N. The name will be 4 blanks if no name was supplied on the header.

6.3.2. Subroutine GTSCAL

SUBROUTINE GTSCAL(S)

GTSCAL gets the scale of the coordinates as it was entered on the SCALE header. The argument is:

REAL S. Will contain the scale.

7. Bimodal assignment control program

7.1. General description of bimodal assignment control program

The bimodal assignment control program (module 4.00 ASSPRO) is called by the supervisor, after the data bank has been opened, to determine which modules to call in order to execute the assignment indicated in COMMON/APARM/ which contains the decoded options and parameters from a $ASSIGN command and its subcommands. The program also accumulates the execution time used in the different modules. The accumulated times are printed at the end of the assignment (module 4.01 PRTASS).

The EMME system is designed to execute five different types of assignment:

- bimodal
- fixed demand transit
- variable demand transit
- fixed demand road traffic
- variable demand road traffic.

The switches in /APARM/ are set by the supervisor so that the bimodal assignment program calls the correct modules. (These switches and the decoded parameters in /APARM/ are not changed by the assignment programs). Some modules are called depending on user specified parameters for a particular type of assignment. Following are lists of the modules which are called for each type of assignment and a general explanation of the methods used. Modules which are only called for certain user specified parameter settings are so noted. Note that other parameters specified in

/APARM/ are handled within the modules listed below or by modules which they call.

The arrows beside the module lists indicate the possible iterations in the bimodal assignment procedure. Another iteration is performed only if the comparisons in TIMCHK (or CAPCHK) are not equal. This is indicated via the argument MODSW which is set to TRUE if any of the new times are significantly different from the previous times (i.e. not equal).

7.1.1. Bimodal assignment modules

5.10	GENER	Generate transit network
→ 4.18	BUSFRQ	Determine or
	UPTLFR	update bus frequency on road links (unless user specified BUSF=NO)
→ 4.21	TIMPED	calculate or
	MODIMP	update transit impedance
6.00	RINIT	Initialize road traffic demand
6.20	TRAFFIC	Perform variable demand road traffic assignment
4.31	TIMCHK	Calculate new transit link times and compare (unless user specified ITLT=0)
5.41	ODTRAN	Calculate current transit demand
5.50	TASS	Perform fixed demand transit assignment
4.61	CAPCHK	Adjust transit line capacity and compare (only if user specified ITCP>0)

The procedure for a bimodal assignment consists of the following steps:

1. Generate transit network from transit line and road network specifications
2. (unless BUSF=NO) Determine bus frequency on road links from transit network.
3. Calculate transit impedance (time) between all OD pairs based on observed road link times or other values as described in module TIMPED.

4. For first iteration only, initialize road traffic demand from OD matrix, auto occupancy and estimated modal split.
5. Perform road traffic assignment, using variable demand algorithm with initial demand as set in step 4,or predicted in the previous execution of this step, to give predicted road link times and volumes and road traffic demand.
6. (If ITLT limit not reached) Calculate new transit link times based on predicted road link times. Increment ITLT. If there is a significant change in any transit link time go to step 3.
7. Calculate transit demand as total demand minus predicted road traffic demand.
8. Assign this transit demand to transit network to give predicted transit line volumes.
9. (If ITCP limit not reached) Calculate new transit line capacity to handle predicted volumes. Increment ITCP. If there is a significant change in any transit line capacity go to step 2.
10. Terminate procedure.

7.1.2. Fixed demand transit assignment modules

5.10	GENER	Generate transit network
5.41	ODTRAN	Calculate transit demand
5.50	TASS	Perform fixed demand transit assignment
4.61	CAPCHK	Adjust transit line capacity and compare (only if user specified ITCP>0)

The procedure for a fixed demand transit assignment consists of the following steps:

1. Generate transit network from transit line specifications, and also road link specifications if these exist.
2. Calculate transit demand from OD matrix and modal split estimate.

3. Assign transit demand to transit network to give predicted transit line volumes.
4. (If ITCP limit not reached) Calculate new transit line capacity to handle predicted volumes. Increment ITCP. If there is a significant change in a transit line capacity go to step 3.
5. Terminate procedure.

7.1.3. Variable demand transit assignment modules

5.10	GENER	Generate transit network
5.30	CALKTR	Calculate component of demand function independent of transit impedance
→5.50	TASS	Perform variable demand transit assignment
4.61	CAPCHK	Adjust transit line capacity and compare (only if user specified ITCP>0).

The procedure for a variable demand transit assignment consists of the following steps:

1. Generate transit network from transit line specifications, and also road link specifications if these exist.
2. Calculate demand function component from constant term plus sum of socionomic terms (file DCOK01), and automobile impedance term (from file RZTIM).
3. Calculate transit impedance (time). Calculate transit demand from total OD matrix using demand function. Assign this demand to transit network to give predicted transit line volumes
4. (If ITCP limit not reached) Calculate new transit line capacity to handle predicted volumes. Increment ITCP. If there is a significant change in any transit line capacity, go to step 3.
5. Terminate procedure.

7.1.4. Fixed demand road assignment modules

5.10	GENER	Generate transit network (only if user specified BUSF=COMP).
4.18	BUSFRQ	Determine bus frequency on road links (only if user specified BUSF=COMP).
6.00	RINIT	Initialize road traffic demand.
6.20	TRAFIC	Perform fixed demand road traffic assignment

The procedure for a fixed demand road traffic assignment consists of the following steps:

1. (If BUSF=COMP) Generate transit network from transit line and road link specifications. This network is used only to compute the bus frequency on road links.
2. (If BUSF=COMP) Determine bus frequency on road links from transit network.
3. Calculate road traffic demand from OD matrix, auto occupancy and estimated modal split.
4. Assign this demand to road network using fixed demand algorithm to give predicted road link times and volumes.
5. Terminate procedure.

7.1.5. Variable demand road traffic assignment modules

5.10	GENER	Generate transit network.
4.18	BUSFRQ	Determine bus frequency on road links (only if user specified BUSF=COMP).
→4.21	TIMPED	Calculate or
	MODIMP	update transit impedance (unless user specified TIMP=PRED)
6.00	RINIT	Initialize road traffic demand.
6.20	TRAFIC	Perform variable demand road traffic assignment.
4.31	TIMCHK	Calculate new transit link times and compare (only if user specified ITLT≠0 and did not specify TIMP=PRED).

The procedure for a variable demand road traffic assignment consists of the following steps:

1. Generate transit network from transit line and road link specifications.
2. (If BUSF=COMP) Determine bus frequency on road links from transit network.
3. (Unless TIMP=PRED) Calculate transit impedance (time) between all OD pairs from transit network based on observed road link times or other values as described in module TIMPED.
4. For first iteration only, initialize road traffic demand from OD matrix, auto occupancy and estimated modal split.
5. Perform road traffic assignment, using variable demand algorithm with initial demand as set in step 4 or predicted in the previous execution of this step, to give predicted road link times and volumes and road traffic demand.
6. (If ITLT limit not reached) Calculate new transit link times based on predicted road link times. Increment ITLT. If there is a significant change in any transit link time go to step 3.

8. Transit section

8.1. General description of transit section

The transit routines can be divided into five groups:

- generation of the transit network (5.10 GENER)
- transit path calculations (5.20 TPATH)
- generation of transit data for road assignment (4.18 BUSFRQ; 4.21 TIMPED)
- iteration control for bimodal assignment (4.21 MODIMP; 4.31 TIMCHK; 4.61 CAPCHK)
- transit assignment (5.50 TASS)

TPATH is called by both TIMPED and TASS. The other routines are called by the bimodal assignment control program (see section 7).

8.1.1. Generation of the transit network

The transit network is represented by a complex list structure located in common area /STRUC/. It is developped in three steps:

- Build the transit network structure representing the in-vehicle links for transit passengers (explicitly) and the transfer links (implicitly) (5.12 BLDNET).
- Interface the transit network with the road network by describing the movement of transit vehicles in terms of road links (5.14 EXPNET).
- Determine travel time on transit links from road link data and from the bus-time/car-time relationship (5.15 TTIME).

Details of the representation of the transit network are given in the description of common area /STRUC/ and of file 5.15 TNET. Since the centroid nodes and the turning road nodes cannot appear on the transit line itineraries, the list of nodes is compressed (1 to NNODET) by substracting NSPENO, as described in the Internal Node Numbers table (section 3.3).

Another function of module 5.15 TTIME is to calculate distances on transit links. In all cases where a straightforward correspondence with a road link can be established, the road link length is used. The length of the other transit links is determined from the node coordinates and the scale. If a distance cannot be calculated, it is set to 99.0 miles.

Modules 5.14 EXPNET and 5.15 TTIME respectively call user-supplied functions TURLIN and BUSROD which are described in section 14 of this manual.

8.1.2. Transit path calculations

Module 5.20 TPATH calculates the tree of shortest paths on the transit network from an origin centroid to all network nodes. It uses the Dijkstra algorithm with a sorted list of candidate nodes kept in a heap structure (the node with the smallest temporary label being on the top of the heap). The tree is transmitted to the calling programs through common area /TRPATH/ and the heap structure is located in common area /HEAP/.

The shortest paths from one centroid to all centroids are calculated in three steps:

1) the calling program (TIMPED or TASS) uses the access links from the origin centroid to initialise the heap and the tree
2) module 5.20 TPATH calculates the tree to all network nodes
3) the calling program uses the egress links to complete the path to each destination centroid.

Shortest paths are calculated in generalised time units where the weight of in-vehicle time is one and the weights of access time and wait time are specified by variables WWALK and WWAIT located in common area /APARM/.

Transfer links do not appear explicitly on the network. The algorithm postulates that all transferring movements are permitted between lines passing at the same node. It should be noted that the algorithm does not permit staying on board at a terminus node (subroutine 5.22 FORLAB and 5.23 REVLAB) and that a transfer cannot be made from a line to the same line (5.21 ACCESS). Transfer time is considered as included in waiting time.

The routines performing the labelling algorithm are:

5.21	ACCESS	5.31	ACCESX
5.22	FORLAB	5.32	FORLAX
5.23	REVLAB	5.33	REVLAX
5.24	LABEL		

The routines managing the heap are:

5.24	LABEL
5.25	DELETE
5.26	INSER
5.27	MODIF

Two types of shortest path calculations can be performed: pure all-or-nothing (ACCESS, FORLAB, REVLAB) or all-or-nothing with diversion over common line sections (ACCESX FORLAX, REVLAX). The selected option is transmitted to subroutine TPATH through its parameter (ACCESS) in the call statement (i.e. CALL TPATH (ACCESS) or CALL TPATH (ACCESX)).

In the case of line diversion, the set of lines leading to a node on the tree is stored in array LIAC in a compacted form (5.32 FORLAX, 5.33 REVLAX). If the lines of a given set are L1, L2 and L3, then LIAC will be LIAC=L1+L2*N+L3*N**2, where N=NLINE+1. The maximum number of lines in a set is specified by parameter NLIMIT located in common area /CSX/.

8.1.3. Generation of transit data for road assignment

Module 4.18 BUSFRQ determines the number of transit vehicles contributing to congestion on the links of the road network. First, a list of lines on the road links is constructed from line itineraries stored on file 514 CORNET. Then the bus volumes, in terms of transit vehicles per hour, are calculated from line headways and are transmitted to module 6.50 BUSIN via file 518 TLFREQ.

Module 4.18 contains another entry point UPTLFR, to permit the updating of the current TLFREQ file after headways have been adjusted. In that case, the list of lines on road links is constructed from data stored on TLFREQ.

Module 4.21 TIMPED calculates the transit impedance matrix which is used by module 6.25 CALCK to calculate the transit component of the demand function for a variable demand road assignment. The procedure is fully described in the module specifications (see Appendix B). In the bimodal assignment, after the road traffic assignment has been completed, the new transit impedances are calculated by calling MODIMP, an entry point in TIMPED. The impedance file is then updated and, as outlined in the next section, a convergence test is made.

8.1.4. Iteration control for bimodal assignment

Module 4.31 TIMCHK calculates new transit link times from the new road link times predicted by a road traffic assignment. The procedure is similar to the one used in the calculation of initial transit times (see 8.1.1). Each new link time is compared with the corresponding link time for the previous iteration by module 4.32 CHANGE. The bimodal procedure converges and is terminated if no significant change is detected in transit link times as indicated in parameter MODSW. Otherwise, new transit impedances are calculated by 4.21 MODIMP.

MODIMP then compares the new impedances with the previous ones. Here again, the bimodal procedure converges and is terminated if no significant change is detected.

The tolerances used in the above comparisons, EPSTIM and TOLERZ respectively, are located in common area /APARM/. The tests are made on the absolute value of the change: ABS ((NEW-OLD)/OLD).

Module 4.61 CAPCHK adjusts line headways in order to provide a capacity that is just above the maximum line volume predicted by a transit assignment. The procedure selects the required headway from a pre-determined list of acceptable headways (array HDWSET in common area /APARM/). If some headways are modified, then the assignment procedure will be repeated (see section 7.1).

8.1.5. Transit assignment

Module 5.41 ODTRAN calculates the OD matrix to be used in a fixed demand transit assignment (file 541 GPQTR). One out of four possible options is selected according to a user-specified parameter SPLT transmitted through common area /APARM/. The options are:

1) $g^{tr}_{pq} = \bar{g}_{pq} - g^{au}_{pq}$

2) $g^{tr}_{pq} = \bar{g}_{pq} - \tilde{g}^{au}_{pq}$

3) $g^{tr}_{pq} = \bar{g}_{pq} * (1-\mu_{pq})$

4) $g^{tr}_{pq} = \bar{g}_{pq} * (1-\mu)$

where

g^{tr}_{pq} = transit trips from p to q (file GPQTR)

g^{au}_{pq} = person trips by car from p to q (file GPQAU)

$\bar{g}_{pq}$ = total person trips from p to q (file PERSPQ)

$\tilde{g}^{au}_{pq}$ = estimated person trips by car from p to q (file MODEST)

μ_{pq} = estimated modal split from p to q (file MODEST)

μ = fixed modal split for all pairs

Variable SPLTV in common area /APARM/ contains the value of μ.

ODTRAN sets the diagonal elements to zero (g^{tr}_{pp}=0.0) and checks for negative entries in the matrix (in which case a message is printed and parameter OK is set to .FALSE.).

Module 5.30 CALKTR calculates the component of the demand function which is independent of the transit impedance: $k01_{pq}+k2_{p}*t^{au}_{pq}$.

The assignment control program (4.00 ASSPRO) will request the execution of either ODTRAN or CALKTR prior to the execution of the transit assignment module, according to the type of transit assignment required (fixed or variable demand).

Module 5.55 TDEM which is called by the transit assignment module TASS returns the vector of transit trips from origin p to all destinations (one row of the OD matrix). It also calculates the total number of transit trips originating from p. Two cases are considered: fixed demand and variable demand transit assignment.

For a fixed demand transit assignment, the trips are read directly from file GPQTR. For variable demand, the term K_{pq}^{tr} (file DCOKTR) and the transit impedance U_{pq}^{tr} are introduced in the demand function to calculate the number of transit trips. One of two functional forms is used (logit or linear), according to parameter DFNC located in common area /APARM/.

Module 5.50 TASS controls the transit assignment. The procedure, which is described in the module specifications (see Appendix B) is summarised by the following steps:

- initialise variables for statistics and predicted volumes
- read in access-egress links
- define weights for walk and wait time
- initialise shortest path algorithm
- request shortest path calculations (CALL TPATH)
- determine trips to be assigned (CALL TDEM)
- assign trips on the network (CALL LOASEC)
- print out and save statistics
- print out and save predicted volumes (CALL FLOOUT).

In the case of a bimodal assignment, shortest paths are read from file TTREES which was produced by module 4.21 MODIMP in the last iteration of the bimodal procedure.

The assignment of the passengers on the network is made by following branches of the shortest path tree from the destination centroid to the origin centroid. The branches of the tree (except for the access and egress links) are defined by a section of a transit line or a section of a set of common lines. Module 5.56 LOASEC accumulates passenger volumes on each of these line sections in arrays DY and DZ of common area /LOADT/. If the branch consists of more than one line, the volume is distributed between lines in proportion to their headways. LOASEC also collects assignment statistics.

Module 5.60 FLOOUT saves predicted volumes on file TFLOW. The volumes are printed for each line, along with operations statistics such as passenger-miles, utilisation factor, etc. Volumes on access-egress links are also printed in this module.

9. Road traffic section

9.1. General description of road traffic section

The road traffic section can be divided into 2 parts:

- initialisation of the demand files (module 6.00 RINIT)
- traffic assignment (module 6.20 TRAFIC)

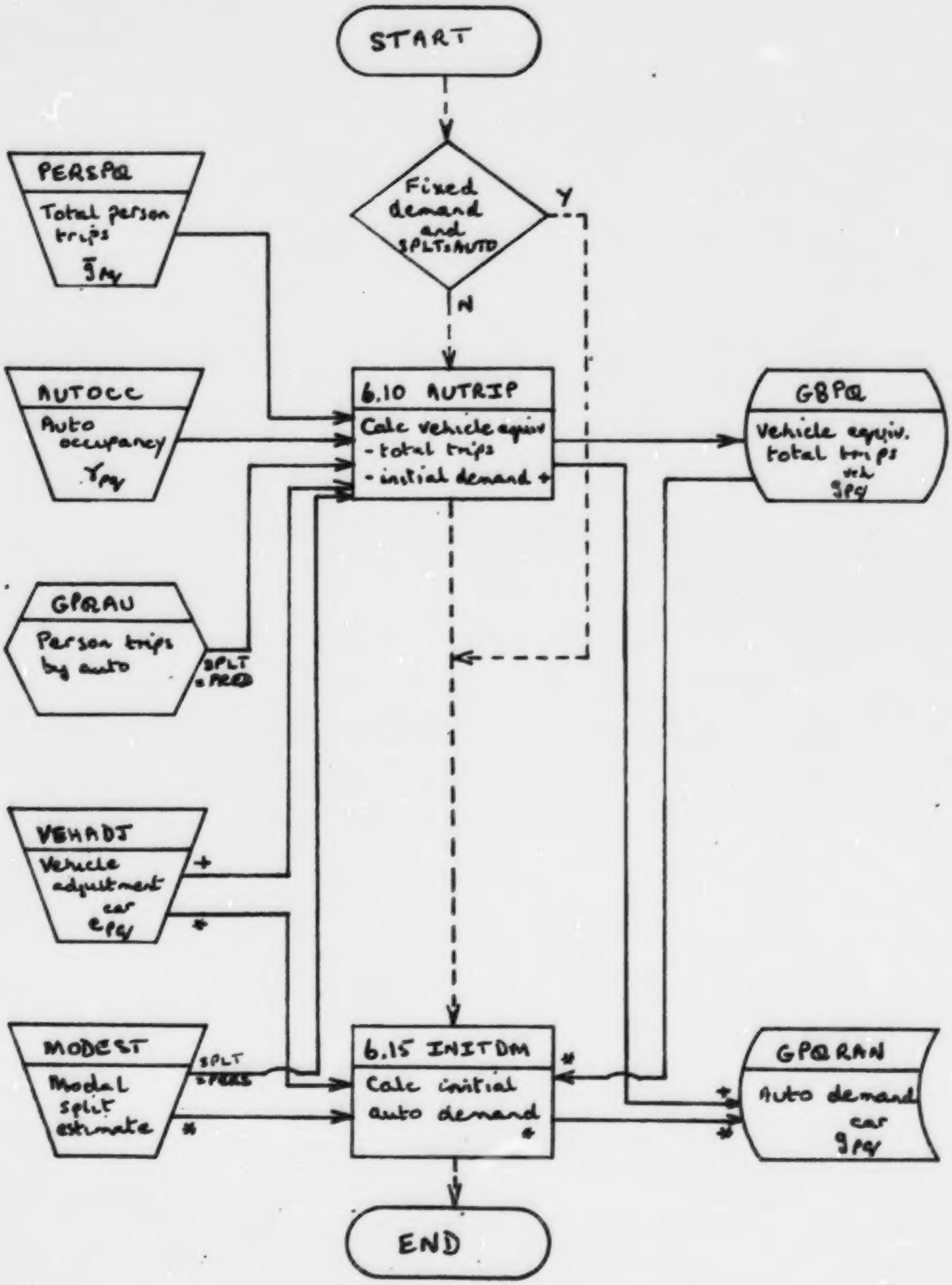
The initialisation of the demand files is performed once for a given assignment (a $ASSIGN command). The traffic assignment is executed once for a fixed demand road traffic assignment but may be executed one or more times within the iterations of a bimodal or a variable demand road traffic assignment (see section 7).

The road traffic assignment calls subroutines supplied by the user in order to calculate the travel time on each link. Paragraph 9.3 contains a list of these routines; more details can be found in Section 14. The handling of a vehicle adjustment matrix and upperbound on the demand are described in paragraph 9.4.

9.1.1. Initialisation of road traffic demand files

This initialisation depends on the parameters SPLT and AOCC specified by the user (or default values). It creates 2 files 601 GBPQ and 602 GPQRAN. GBPQ contains the auto equivalent of the total trips ($\bar{g}_{pq}^{veh}$) which is calculated in module 6.10 AUTRIP (called from 6.00 RINIT) from the OD matrix file 221 PERSPQ and auto occupancy specified by parameter AOCC. GPQRAN contains the auto equivalent of the initial estimate (bimodal or variable demand) or actual number (fixed demand) of trips by auto (g_{pq}^{car}) calculated according to the value of parameter SPLT (modal split). If the modal split is specified as the number of persons, GPQRAN is created in module 6.10 AUTRIP using the auto occupancy specified by parameter AOCC. If the modal split is specified as a percentage of the total trips or as the number of trips by auto, GPQRAN is created in module 6.15 INITDM (called from 6.00 RINIT). In either case if a vehicle adjustment matrix is to be used, indicated by parameter VEAD=YES, this matrix is read and added to the number of trips by auto in the module which creates GPQRAN.

A flowchart representing the code contained in 6.00 RINIT and the files used by the modules it calls is shown in Figure 9-1.


START
Fixed demand and SPLT=AUTO
Y
N
PERSPQ
Total person trips
$\bar{g}_{pq}$
AUTOCC
Auto occupancy
γ_{pq}
GPQAU
Person trips by auto
SPLT =PRED
VEHADJ
Vehicle adjustment car
e_{pq}
+
*
MODEST
Modal split estimate
SPLT =PERS
*
6.10 AUTRIP
Calc vehicle equiv
- total trips
- initial demand +
GBPQ
Vehicle equiv. total trips
veh
g_{pq}
6.15 INITDM
Calc initial auto demand
*
*
GPQRAN
Auto demand car
g_{pq}
+
*
END

+ if SPLT = PERS or PRED
* if SPLT = AUTO or PCNT or x.xx

Figure 9-1: Flowchart of module 6.00 RINIT. Initialize road traffic demand

9.1.2. Road traffic assignment

A road traffic assignment (module 6.20 TRAFIC) consists of setting up the data describing the road network, obtained from the data bank, into COMMON areas and then assigning the auto trip demand, calculated as described in 9.1.1, according to the iterative algorithm described in module 6.60 RASS. The resulting link times and volumes in a COMMON area are saved by module 6.80 RASSAV.

In addition, if the road traffic assignment is with variable demand, module 6.60 RASS will use the variable demand algorithm. This requires the demand function component K_{pq}^{au} which is calculated by module 6.25 CALCK prior to entering 6.60 RASS. The variable demand traffic assignment algorithm has been described by Sang Nguyen in "Procedures for Equilibrium Traffic Assignment with Elastic Demand", publication # 39, Centre de recherche sur les transports, Université de Montréal, pages 3-8. This description is reproduced in Appendix E.

The final contents of file GPQRAN are transformed to person-trips and saved on file GPQAU by module RASSAV. In the case of variable demand this gives the number of person-trips by auto predicted. For fixed demand, it gives the number of person-trips by auto assigned, that is as calculated for the initial demand.

A flow chart representing the code contained in module 6.20 TRAFIC and the modules it calls together with the files used by these modules is shown in Figure 9-2.

* indicates 'for variable demand only'

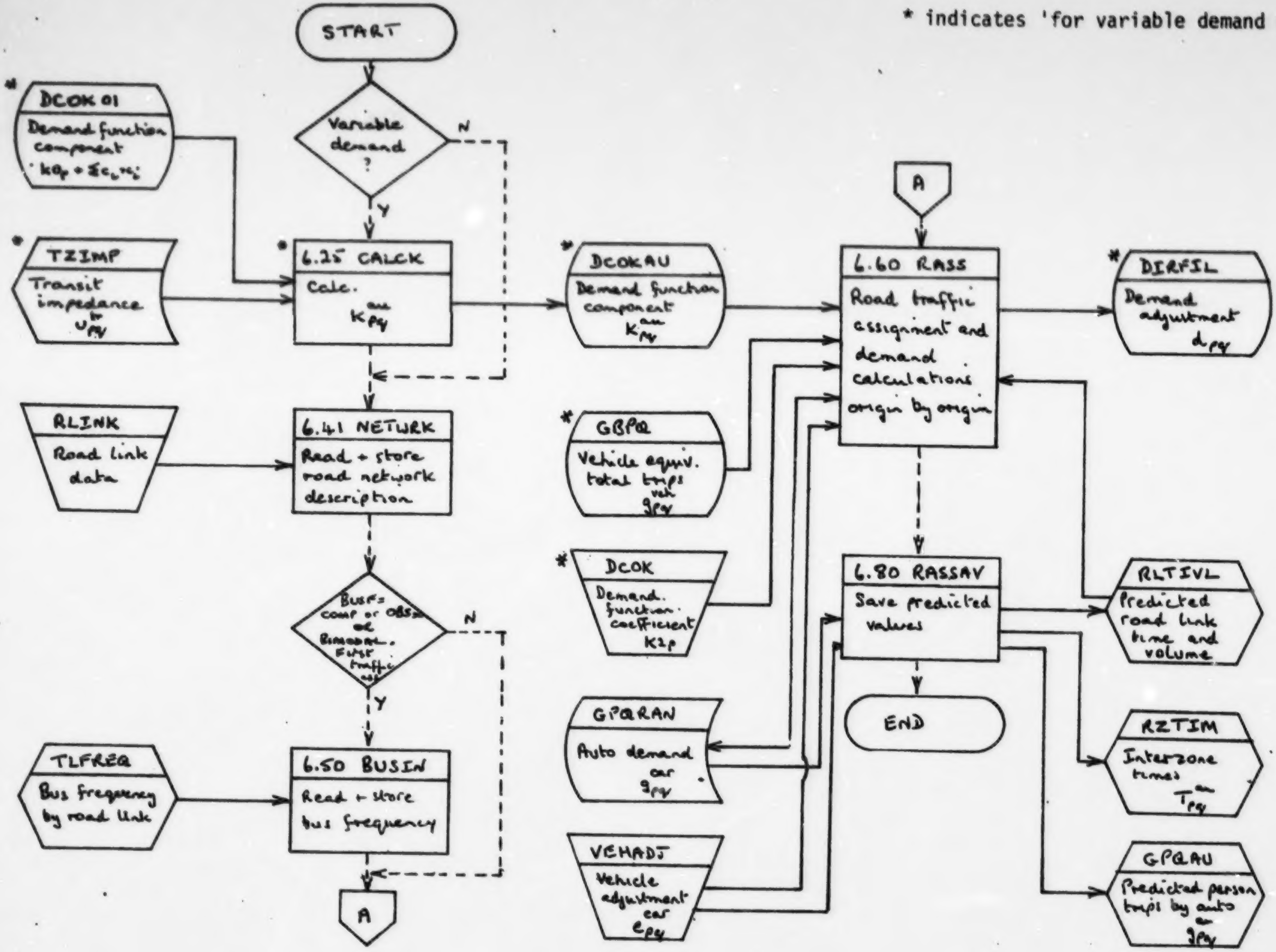

START
Variable demand ?
N
Y
*
DCOK 01
Demand function component $k a_p + \Sigma c_i x_j$
*
TZIMP
Transit impedance U_{pq}
*
6.25 CALCK
Calc. K^{au}_{pq}
RLINK
Road link data
6.41 NETWRK
Read + store road network description
BUSF= COMP or OBS= or BIMODAL. First traffic ass
N
Y
TLFREQ
Bus frequency by road link
6.50 BUSIN
Read + store bus frequency
A
*
DCOKAU
Demand function component K^{au}_{pq}
*
GBPQ
Vehicle equiv. total trips g^{veh}_{pq}
*
DCOK
Demand. function. coefficient $K2p$
GPQRAN
Auto demand car g_{pq}
VEHADJ
Vehicle adjustment car e_{pq}
A
6.60 RASS
Road traffic assignment and demand calculations origin by origin
6.80 RASSAV
Save predicted values
END
*
DIRFIL
Demand adjustment d_{pq}
RLTIVL
Predicted road link time and volume
RZTIM
Interzone times T^{au}_{pq}
GPQAU
Predicted person trips by auto g^{au}_{pq}

Figure 9-2: Flow chart of module 6.20 TRAFIC. Road traffic assignment with variable or fixed demand

9.2. Road traffic section: Use of COMMON areas.

In the following description an iteration refers to an iteration within the road traffic assignment procedure (module 6.60 RASS).

The road traffic section uses COMMON areas as follows:

COMMON area	Use
SUPCOM	Transmits switch which controls printing of debugging output (DEBUG). Set by supervisor from a $DEBUG command.
UNIT	Contains logical unit numbers of files.
NETSIZ	Specifies size of current road network, Initialised by opening data bank (module 3.21 OPBAN called by supervisor).
APARM	Transmits assignment parameters which user may vary, decoded by supervisor.
RNET1	Filled with road network description (module 6.41 NETWRK) used by traffic assignment (module 6.60 RASS and modules it calls).
RNET2	Used during assignment to contain the predicted link times and volumes. Updated each iteration. Also contains demand for current origin, read origin by origin from file GPQRAN during each iteration and updated if assignment uses variable demand.
RVAR1	Contains intermediate values calculated during each iteration.
RVAR2	Used only for a variable demand assignment. Contains values needed for the demand function calculation, for the current origin. These values are read origin by origin from files DCOKAU, GBPQ and DCOK during each iteration.

RBUS	Only used if BUSF=YES. Contains bus frequency on each road link. Read in from file TLFREQ at beginning of assignment and used in travel time calculations.
REXOD	Contains vehicle adjustment matrix for the current origin. Initially set to zero and changed only if VEAD=YES. Read in from file VEHADJ origin by origin during initialisation of demand files during each iteration, and when GPQRAN is saved (module 6.80 RASSAV).
REP	Page number for report titles is set and incremented for assignment reports.
IX	Work area used to contain random file index for file GPQRAN.

Common areas RNET1, RNET2, RVAR1, RVAR2, RBUS, REXOD are set up and used by the road traffic section. The other areas transmit data between other parts of the EMME system and the road traffic section or vice-versa.

9.3. Road traffic section: User specified routines

The road traffic section contain the following calls to user supplied routines.

User supplied routine	Called from
TURLIN	6.68 MOES
LINKIN	6.41 NETWRK
FINDLC	6.71 USECST
	6.72 TLCST

Details of the routines can be found in section 14.

9.4. Road traffic section: Vehicle adjustment matrix and upper bound on demand

The vehicle adjustment matrix (g_{pq}^{e}) represents the number of vehicles between each OD pair which are not included in the OD matrix to be assigned but which must be included when considering the congestion on the links of the road network. g_{pq}^{e} is added to the vehicle equivalent of trips by auto on file GPQRAN in module 6.10 AUTRIP or 6.15 INITDM (see 9.1.1). g_{pq}^{e} is subtracted from the number of trips on GPQRAN before it is converted to person-trips and saved on GPQAU in module 6.80 RASSAV. g_{pq}^{e} is also subtracted from the total number of vehicle trips (file GPQRAN) before the number of vehicle trips is used in demand function calculations in modules 6.73 DEMCST and 6.74 TDCST.

The variable demand road traffic assignment algorithm involves the calculation of the inverse of the demand function i.e. the automobile impedance as a function of the demand. If one demand function is calibrated for several OD pairs it may be possible to predict a demand which gives a negative automobile impedance. In order to prevent this the demand is limited to an upper bound which is the demand predicted for an automobile impedance of zero. This upper bound is calculated in 6.60 RASS in order to reduce the initial demand if necessary and is used in 6.61 LOADTR to limit the demand adjustment to the upper bound.

10. Demand function section

10.1. General description of demand function section

The routines in this section are used only if the user requests a bimodal or variable demand assignment via a $ASSIGN command.

The demand function in the EMME system gives the fractions of the total trips which will be made by auto or by transit. The function may be linear or a logit function (parameter DFNC=LINE or LOGI) but in both cases is a function of the following variables

- socioeconomic variables (x_{ip}, x_{iq} and x_{ipq})
- transit impedance (T^{tr}_{pq})
- auto impedance (T^{au}_{pq})

There is also a constant term (k_0) and each variable has a coefficient. Figure 10-1 shows the form of the demand functions and explains the symbols used.

In a variable demand transit assignment, the demand function is used to calculate the number of trips by transit (demand) for given values of the socioeconomic variables and the calculated values of transit and auto impedances.

In a bimodal assignment or a variable demand road assignment, both of which use the variable demand road traffic assignment algorithm, the inverse of the demand function is used, that is value of the automobile impedance predicted for the current value of the demand between each OD pair (see Section 9).

Linear function:

$$g_{pq}^{au} = \bar{g}_{pq} \cdot (k_0 + \sum_i c_i x_i + k_1 T_{pq}^{au} + k_2 T_{pq}^{tr})$$

Logit function:

$$g_{pq}^{au} = \bar{g}_{pq} / [1 + \exp (k_0 + \sum_i c_i x_i + k_1 T_{pq}^{au} + k_2 T_{pq}^{tr})]$$

where $T_{pq}^{tr} = T_{pq}^{tr}(v) + w_1 T_{pq}^{tr}(acc) + w_2 T_{pq}^{tr}(wait)$

$T_{pq}^{tr}(v)$ = In vehicle transit time from p to q

$T_{pq}^{tr}(acc)$ = Transit access (egress) time for trips from p to q

$T_{pq}^{tr}(wait)$ = Transit waiting time for trips from p to q

p = origin zone

q = destination zone

g_{pq}^{au} = demand by auto from p to q

$\bar{g}_{pq}$ = total demand from p to q

k_o, k_1, k_2, w_1, w_2 are coefficients obtained by calibration and can be different for each p

c_i = coefficients of socioeconomic variables obtained by calibration and may be different for each p and q

Figure 10-1a: Demand function forms and symbols

Intermediate values calculated are:

$$K_{pq}^{au} \doteq k_0 + \sum_i c_i x_i + k_2 \ T_{pq}^{tr}$$

$$K_{pq}^{tr} = k_0 + \sum_i c_i x_i + k_1 \ T_{pq}^{au}$$

which give $g_{pq}^{au} = \bar{g}_{pq} \ . \ (K_{pq}^{au} + k_1 \ T_{pq}^{au})$ Linear function

$g_{pq}^{tr} = \bar{g}_{pq} \ . \ (1-K_{pq}^{tr} - k_2 \ T_{pq}^{tr})$ " "

$g_{pq}^{au} = \bar{g}_{pq} \ / \ [1 + \exp \ (K_{pq}^{au} + k_1 \ T_{pq}^{au})]$ Logit function

$g_{pq}^{tr} = \bar{g}_{pq} \ . \ \{1 - 1 \ / \ [1 + \exp \ (K_{pq}^{tr} + k_2 \ T_{pq}^{tr})]\}$

The function inverses are:

$u_{pq}^{au} = \frac{1}{k_1} \ (- K_{pq}^{au} + \frac{g_{pq}^{au}}{\bar{g}_{pq}})$ Linear function

$u_{pq}^{au} = \frac{1}{k_1} \ [- K_{pq}^{au} + \ell n \ (\frac{\bar{g}_{pq}}{g_{pq}^{au}} - 1)]$ Logit function

Figure 10-1b: Demand function forms and symbols (continued)

The values for the socioeconomic variables are defined by the user and entered into the data bank as are the values of all the coefficients. Transit impedance and auto impedance are computed within the assignment programs (refer to sections 8 and 9).

The demand function matrix (1 value for each OD pair) is computed in 3 components:

- constant term plus socioeconomic variable terms
- transit impedance term
- auto impedance term.

The first component is computed in module 7.10 DCOS01 which is called by the supervisor when the first $ASSIGN command for a bimodal or variable demand assignment is encountered. DCOS01 is called again only if such an $ASSIGN command is encountered following data bank modifications (including using a scenario). The first component is written as a matrix on file DCOK01. The transit impedance term is added by module 6.25 CALCK which is called from 6.20 TRAFFIC for a road assignment with variable demand (applies also to bimodal assignments). This term is also calculated in module 5.50 TASS, called from 4.00 ASSPRO, for a variable demand transit assignment.

The automobile impedance term is added explicitly by module 5.30 CALKTR, called from 4.00 ASSPRO for a variable demand transit assignment. This term is handled implicitly within the algorithm for a variable demand road traffic assignment (see Section 9). The following table summarizes the modules which calculate the components for different types of assignment.

Component / Assignment	Constant + socionomic	Transit impedance	Auto impedance
bimodal; road, variable	7.10 DCOSO1	6.25 CALCK	6.73 DEMCST* 6.74 TDCST*
transit, variable	7.10 DCOSO1	5.50 TASS	5.30 CALKTR

* implicitly

11. Graphics programs

11.1. General description of graphics programs

The computer graphics package developed for EMME consists of four independent programs designed to plot the vehicle and transit networks and the predicted vehicle and transit link flows over a specified region of the network. The structure of this subsystem and its inter-relation with the EMME system is illustrated in fig. 11-1. Network descriptions and node coordinates are obtained from the EMME data bank; predicted link flows from intermediate output files; additional information relating to plot characteristics is contained in two user supplied files.

The node coordinates are obtained by geo-coding maps of the network covering the study area. If the scale and orientation differs by map (e.g. the regional network map may be at a larger scale than the downtown map), coordinate transformations must be carried out to ensure that the coordinate information entered into the EMME data bank has a single system of reference. Since the user can specify the scale and orientation of the area to be plotted, this allows the plotting of any subarea of the network and in particular corridors which lie in more than one map. The scale of the coordinates in the data bank is obtained by calling the EMME library routine GTSCAL: the variable returned, UNIT, designates the number of data bank units per mile (coordinates in data bank are integer variables). The plot area is contained within a rectangle inclined at an angle Θ to the x-axis of the data bank coordinate system (Fig. 11-2.). It is completely defined by the specification of four nodes, one on each side of the rectangle, and Θ. A rectangle internal to this region, defined in a

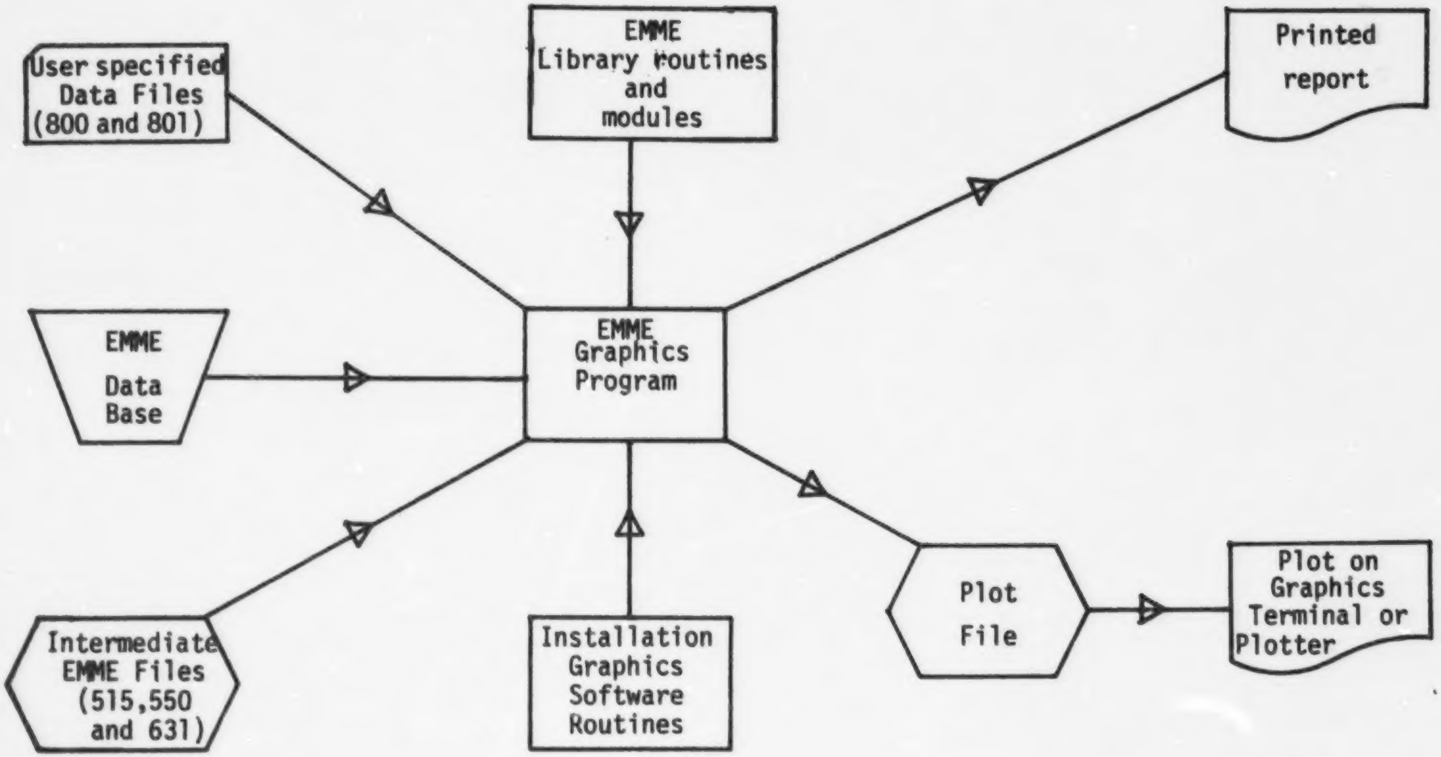

User specified Data Files (800 and 801)
EMME Library routines and modules
Printed report
EMME Data Base
EMME Graphics Program
Intermediate EMME Files (515,550 and 631)
Installation Graphics Software Routines
Plot File
Plot on Graphics Terminal or Plotter

Fig. 11-1. - Diagrammatic representation of interrelation between graphics programs and EMME

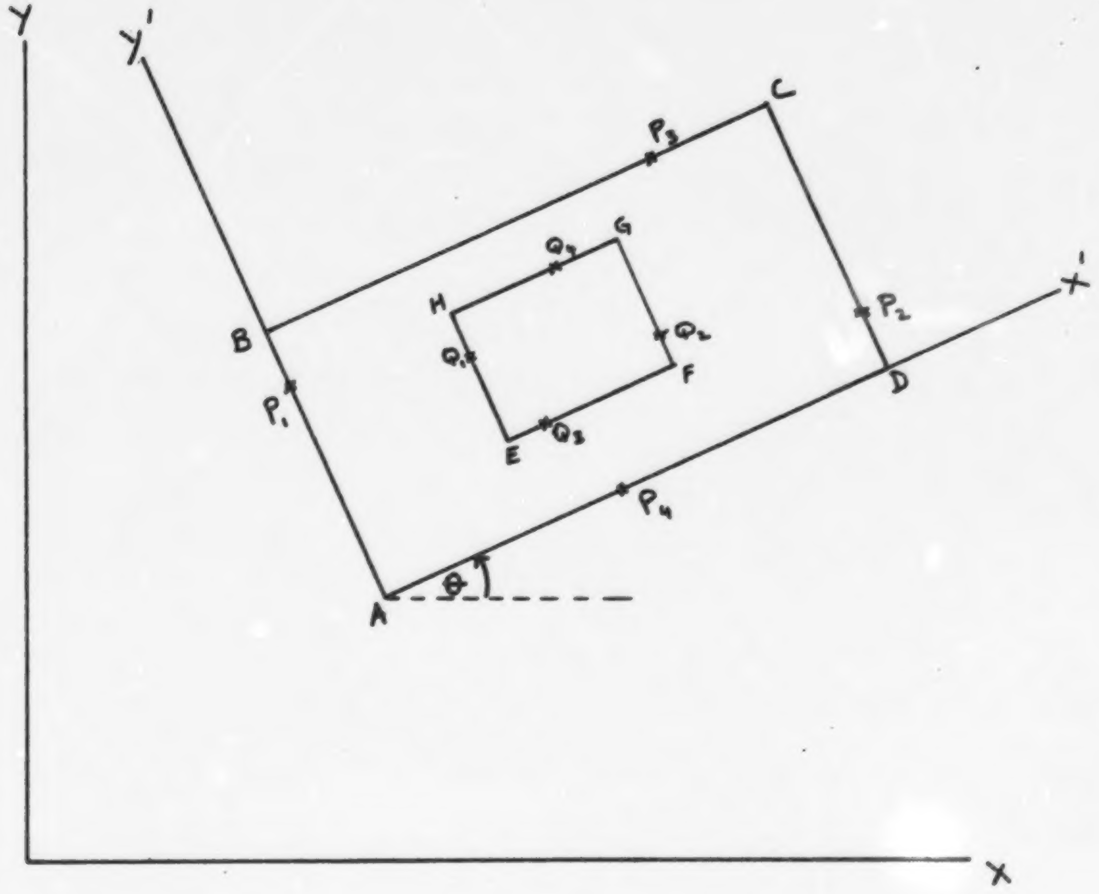

Y
Y'
C
P3
G
Q4
X'
H
P2
B
Q2
Q1
F
D
P1
Q3
E
P4
θ
A
X

Fig. 11-2. The points P_i, i=1, ..., 4, define the rectangle ABCD to be plotted.

Q_i, i=1, ..., 4, define rectangle EFGH to be omitted.

Θ is the angle of rotation, counterclockwise sense of the plot x-axis with respect to the x-axis of the data bank coordinate system.

similar manner, may be optionnaly omitted from the plot. This allows for the separate plotting of the regional and downtown portions of the networks which is desirable since the latter usually requires a larger scale.

The plot size is determined by the nodes P_1, P_2, P_3, P_4 and the plot scale which is user specified. The plot height AB is limited by the width of the plot paper. 2½ inches at the top of the plot is reserved for printing a title, the scale (and volume scale in the case of volume plots), and a north direction arrow. Characteristics of each plot are outlined separately below.

11.1.1. Vehicle network (Fig. 11-3.)

A pair of connected nodes is joined by a solid line with a directional arrow located at its midpoint if the corresponding link is one-way. Node numbers are drawn parallel to the x-axis. Centroid connectors are indicated as dashed lines.

11.1.2 Transit network (Fig. 11-4.)

This has the same characteristics as the vehicle network plot with the additional information of the bus line numbers serving each link included (if the links are long enough to contain them). Also nodes lying on express bus routes are doubly defined. Optionally, the routes for a subset of bus lines may be plotted.

11.1.3. Vehicle assignment (Fig. 11-5.)

Connected nodes are joined by dashed lines to indicate the network structure. A link volume is represented by a solid rhomboidal segment, the width of the segment indicating the volume assigned to the link. The traffic volume is printed at the midpoint of the segment. On centroid connectors and "dummy" links, volume numbers only are plotted.

11.1.4. Transit assignment (Fig. 11-6.)

This is similar to the vehicle assignment plot with the additional option of plotting the link volumes for a subset of bus lines. Link volumes contributed by express lines are included with those of other lines using the same physical link. On these links only the lower node numbers are used for link identification.

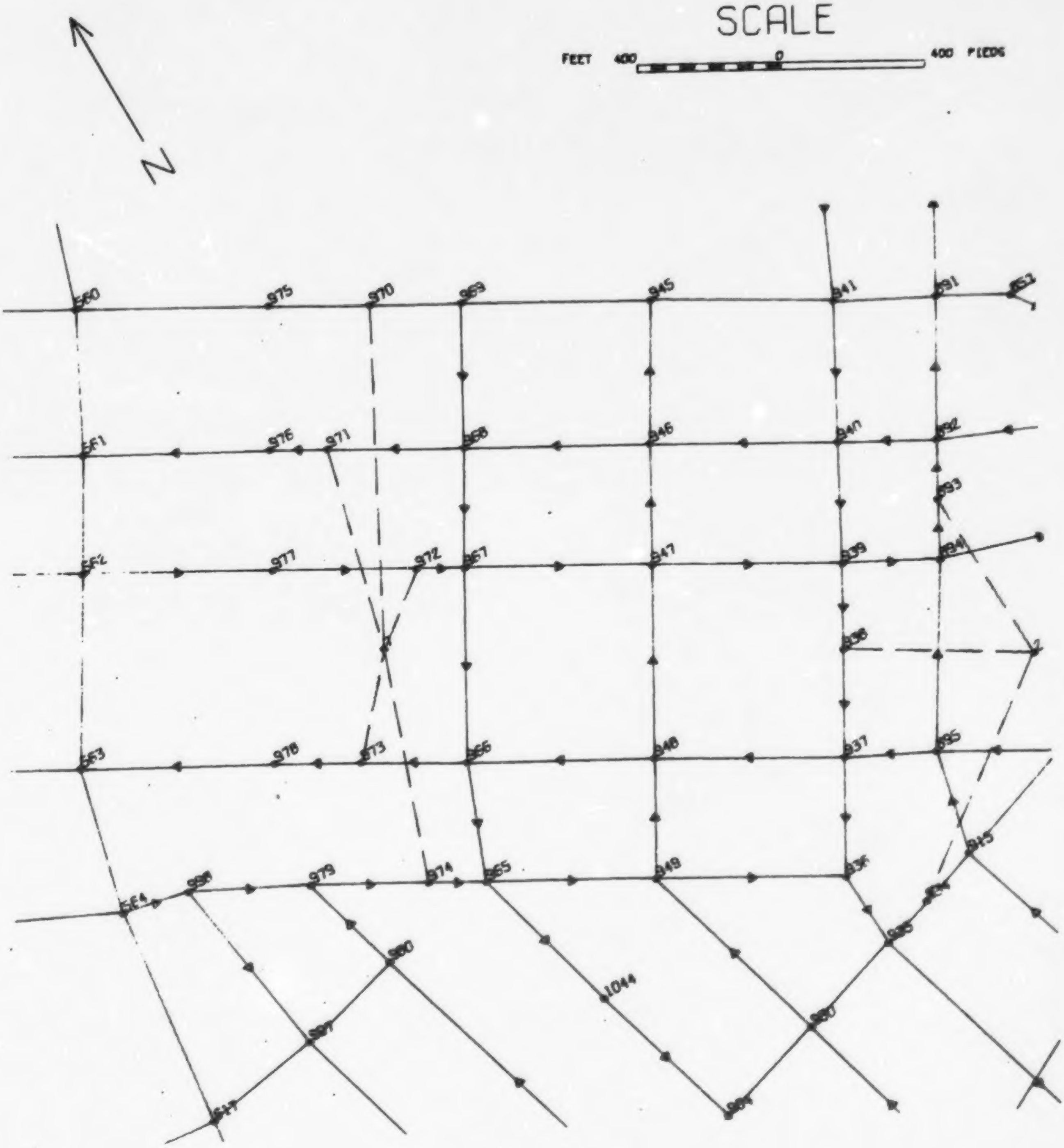
11.2. Structure of programs

The modules 8.21, 8.41, 8.61 and 8.81 (AUNETP, AUVOLP, TRNETP, TRVOLP) comprise the MAIN routine of each plot program (vehicle network, vehicle volume, transit network and transit volume respectively). The modules in the range 8.21 - 8.39 are subroutines required by AUNETP; 8.41 - 8.59 are required by AUVOLP, etc. Modules in the range 8.01 to 8.15 are subroutines required by more than one of the plot programs. Specifically modules 8.01 - 8.04, 8.09, 8.11 are required by each program and all the modules in the range 8.01 - 8.15 are required by the volume plot programs. Some EMME library subroutines and modules are also required to execute the plot programs.

11.3. Possible user modifications of programs

The programs are written as far as possible in standard fortran and were designed to function on the CDC system at the Université de Montréal. Their transfer to another system is straightforward though may require minor fortran modifications and changes in call statements to the system plot routines. For the convenience of a potential user we list the anticipated changes below.

VEHICLE NETWORK



SCALE
FEET 400
0
400 FEET
N
960
975
970
969
945
941
991
953
961
976
971
968
946
940
992
993
962
977
972
967
947
939
994
938
963
978
973
966
948
937
995
964
958
979
974
965
949
936
915
935
980
1044
997
917

Figure 11-3.

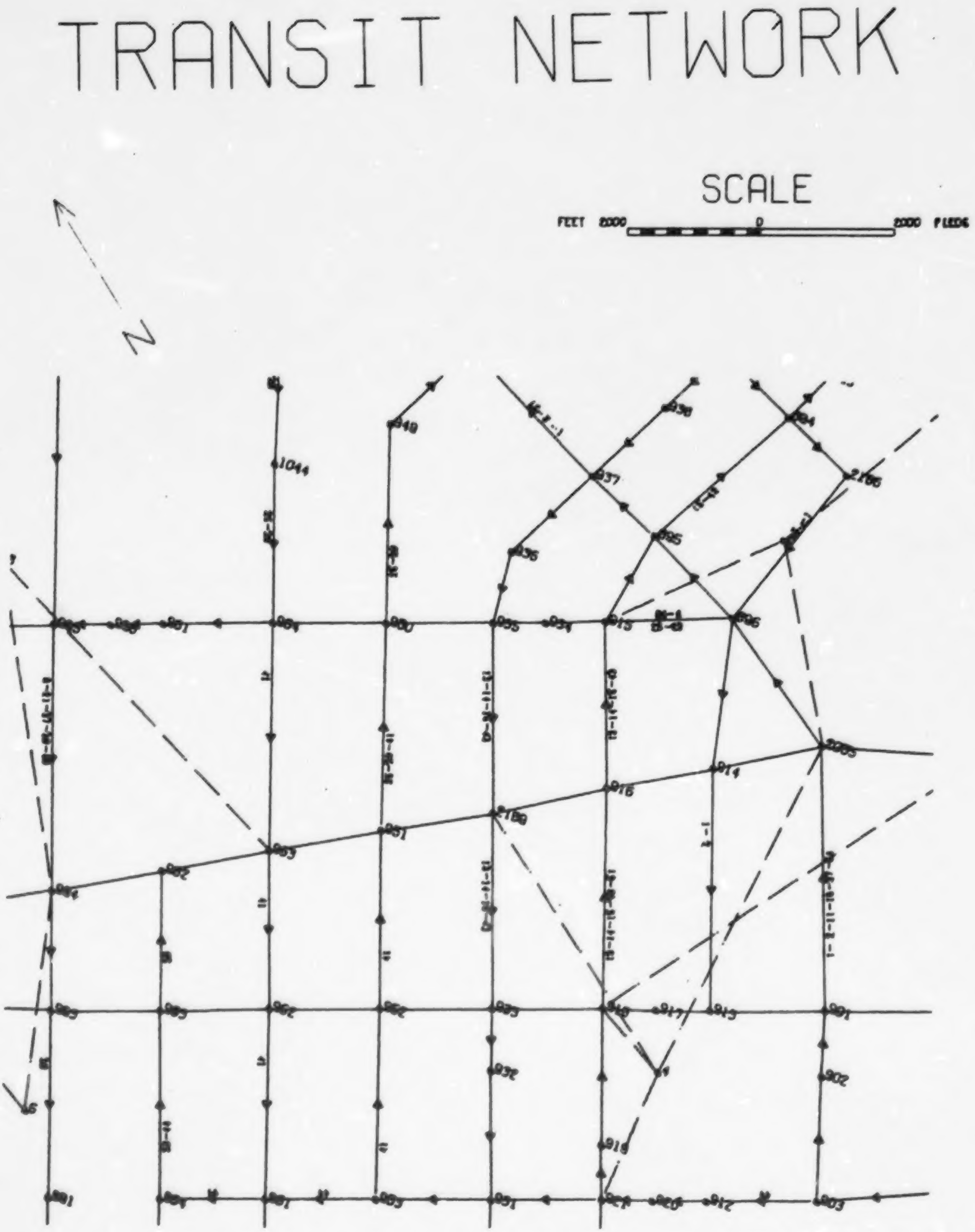

TRANSIT NETWORK
SCALE
FEET 2000
0
2000 PIEDS
N
938
934
949
1044
937
2186
935
936
915
916
914
2005
903
931
902
913
917
919
932
1

Figure 11-4.

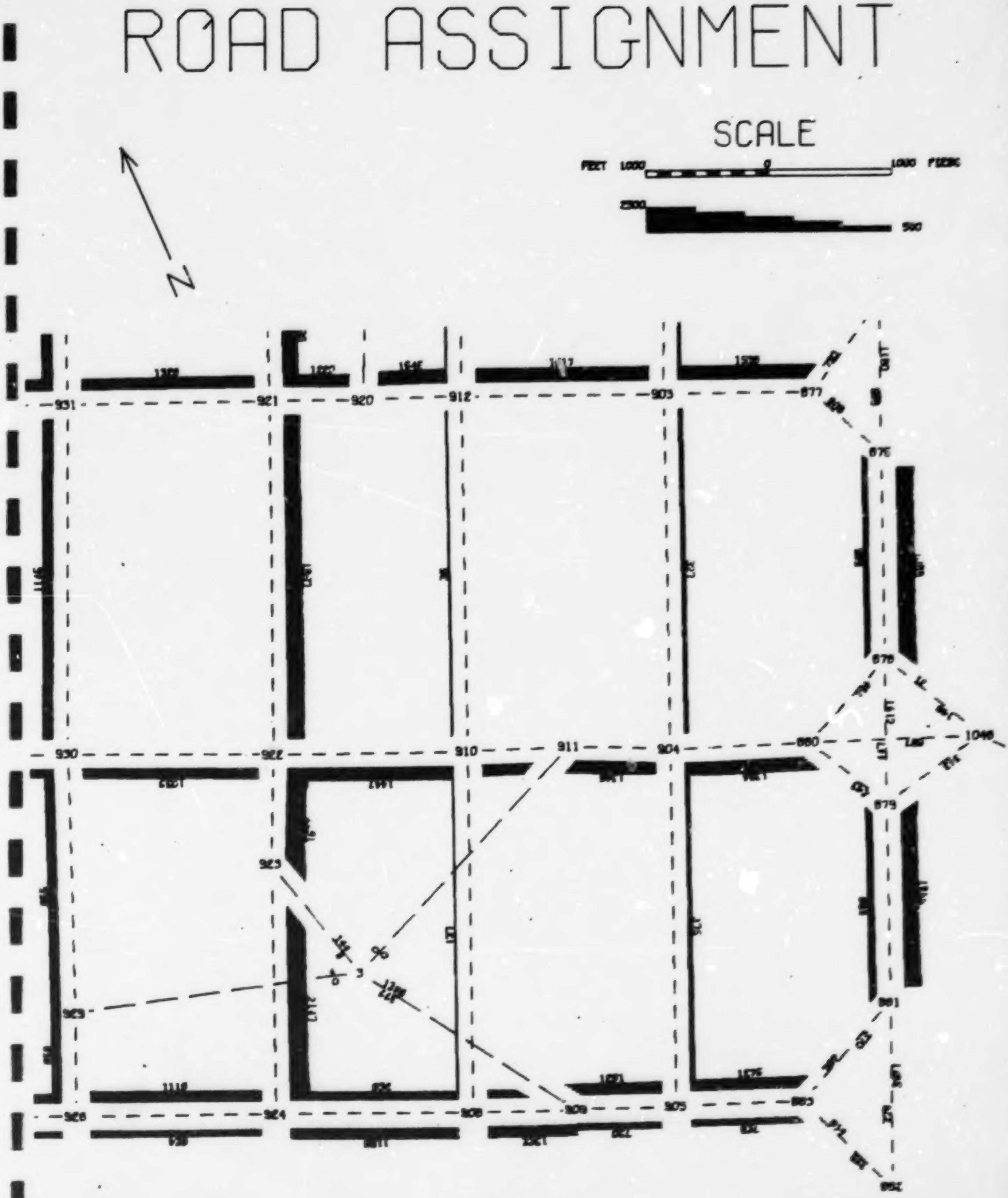

ROAD ASSIGNMENT
SCALE
FEET 1000
0
1000
2500
500
N
931
921
920
912
903
877
876
878
880
1040
879
881
883
852
930
922
910
911
904
923
929
926
924
908
909
905

Figure 11-5.

TRANSIT ASSIGNMENT

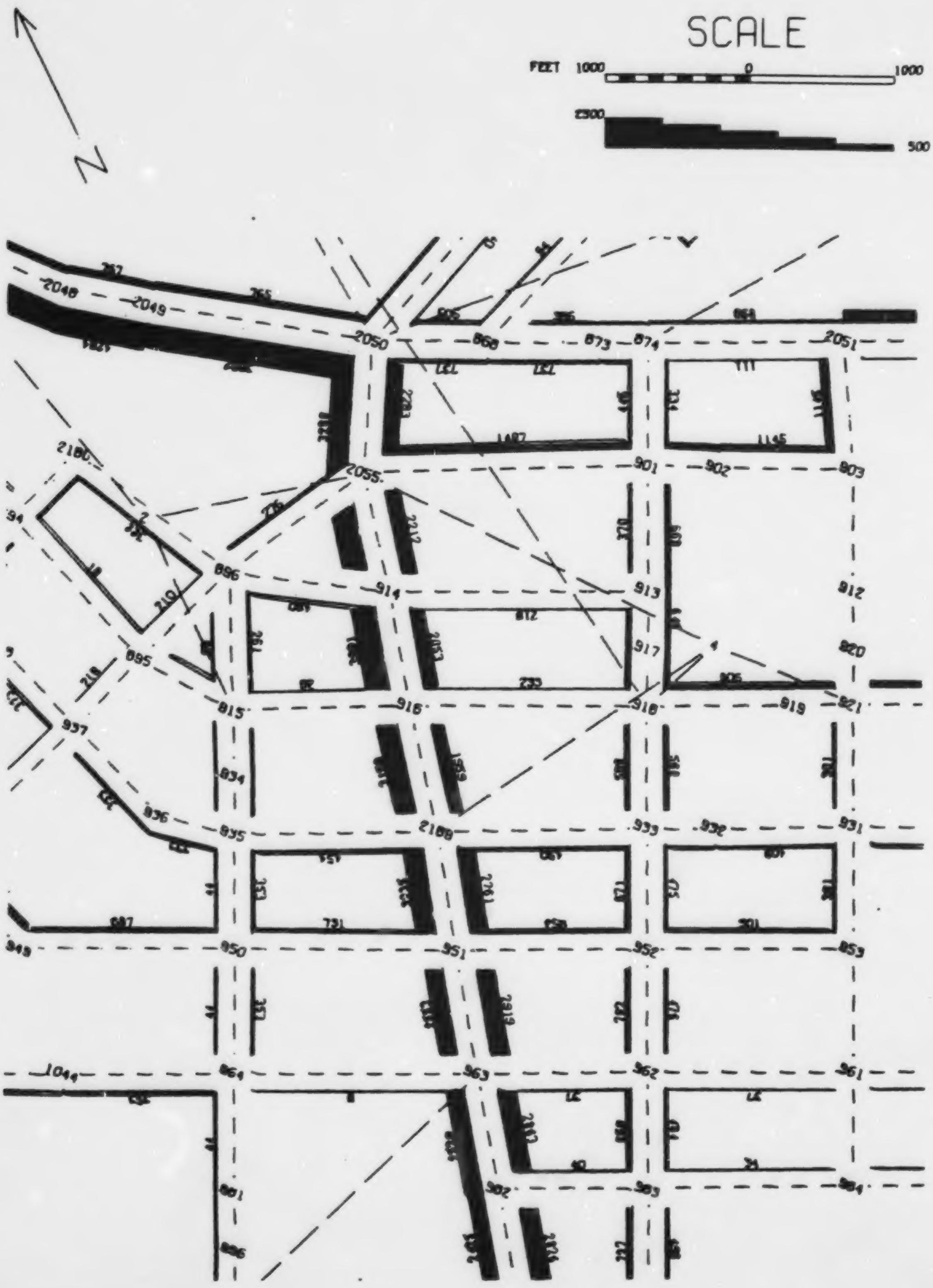

SCALE
FEET 1000
0
1000 PIEDS
2500
500
N
2048
2049
2050
868
873
874
2051
2055
2180
901
902
903
896
914
913
912
895
917
920
815
916
918
919
921
937
834
836
835
2180
933
932
931
949
950
951
952
953
1044
964
963
962
961
981
982
983
984
986
1487
1145
2212
2053
2161
2919

Figure 11-6.

11.3.1. CDC specific fortran

- use of double quotes (") to contain a character string in some format statements.
- two occurences of A10 in format statements (modules 8.01 and 8.43).
- the link line in each main routine (modules 8.21, 8.41, 8.61, 8.81) identifies the I/O units used and is not standard. (I/O units are specified in the BLOCK DATA subprogram module 8.09).

11.3.2. Numerical information

Some data which may be system specific is defined in the BLOCK DATA subprogram (module 8.09). These are denoted by the following variables:

WSYM - ratio width/height of plotted character using system plot routines SYMBOL and NUMBER (see below for description of these routines).

YMAX - maximum plot height (inches) permitted by plotter.

INFINI - value of 'infinity'. This must have the same value as that specified in the BLOCK DATA subprogram of the EMME supervisor. Data bank node coordinates are initialized to this value and its purpose in the plot programs is to determine which nodes have not been allocated coordinates.

XINFIN - same value as INFINI but as a real variable.

Input file numbers PSPEC, PTITDT, RLTIVL, TNET, TFLOW are specified in the BLOCK DATA subprogram and are transferred to other subroutines through COMMON blocks UNIT, BANK and PLUNIT.

11.3.3. CDC library routines

Subroutine DATE (returns date at which run is executed) is called in the main routine of each plot program.

11.3.4. CDC plot software

There may be modifications in the call statements for some system plot subroutines. The system plot software routines used are listed below and descriptions from the CDC plot manual are included to facilitate modifications.

CDC Plot Routine	
LIMITE(X,H,Y)	Limits the region of the pen to ranges 0"-X" on x-axis and 0"-Y" on y-axis. The value of H is 10HPERMISSION.
ENDPLT	Subroutine indicating termination of plots.
PLOT(X,Y,IPEN)	The pen moved to point (X,Y). If IPEN=2, pen is down; if IPEN=3, pen is up.
NUMBER(X,Y,H,N,A,F)	Plots the integer number N at angle A (degrees) to x-axis such that the lower left corner of the imaginary rectangle which contains the first digit (before rotation) is the point (X,Y). H is the height in inches of the characters; the width is (6/7)H. (The radio width/height is denoted by WSYM in the plot programs and its value is specified in the BLOCK DATA subprogram). F contains the format (in the fortran sense) in which the number is to be plotted (e.g. F=5HE10.2).
SYMBOL(X,Y,H,K,A,L)	X,Y,H and A have the same interpretation as for NUMBER. If alphanumeric characters are being plotted, K may be: - a Hollerith constant (nH...) - a vector containing the characters.

In this case L designates the number of characters. (N.B. on the CDC a word may contain 10 characters). If a special symbol is being plotted, L is set to -1 to indicate that the pen is up in moving to (X,Y). K is an integer indicating the special symbol desired. The special symbols used are:

- circle where H denotes diameter and centre is at (X,Y),K=1;
- arrow head with median at (X,Y),K=2;
- asterisk, centre at (X,Y),K=11.

11.3.5. Program limitations

There is one limitation in the transit network plot program (modules 8.61 - 8.79) not inherent in EMME, due to the storage limitations on the CDC; namely that the number of bus lines must be less than 100. This limitation occurs because the bus line numbers for each link are stored in packed form in the array BLINE and it is assumed that each line number in the internal representation does not contain more than two digits. This could be extended to three digits (permitting up to 1000 different bus lines) by making the appropriate changes in the computer code; i.e. in subroutines where the line numbers are stored or unpacked, and in the dimensions of the integer variable BLINK in COMMON block /LINK/. For example, at present BLINK is dimensioned so that there are four words for each link and each word can contain four 2-digit numbers permitting a maximum of 16 bus lines to be plotted along the link median (Fig. 11-4.). The manner in which the line numbers are stored in BLINE is explained in the description of module 8.63 (ROUTE).

12. REPORT PROGRAMS

12.1. General description

The report program section covers the report programs that are executed as a result of $REPORT subcommands. These report programs may be executed even if an assignment has not been performed in the same execution of the EMME system provided that the necessary files have been saved from the assignment for which the reports are required. Other reports are produced during an assignment.

The $REPORT subcommands are decoded by the supervisor in module 1.70 DECREP which calls the report program indicated by each subcommand. Currently the following subcommands are accepted and the modules indicated are called.

Subcommand	Module called
DEMAND	6.92 PRTGPQ
LINK	6.94 PRTLNK
AUTO TIME DISTRIBUTION	6.96 RTRIPD

The table containing the subcommands in DECREP may be expanded to include other subcommands. The format of the table can be found in the description of module 1.12 ANWORD in Appendix B.

The function of the individual subcommands is described in the EMME User's Guide. Description of the modules called are contained in Appendix B.

Since the report programs depend on certain parameters which the user may specify in the $ASSIGN subcommand PARAMETERS, $REPORT accepts this subcommand to set values in COMMON/APARM/ if there is no preceding assignment in this execution of EMME. Values in /APARM/ may also be changed via the PARAMETERS subcommand. Decoding of this subcommand is as described for module 1.62 DECPAR.

13. SEGMENTED (OVERLAY) LOAD

Since the EMME system design is hierarchical, as described in section 3.1, the whole system need not be in core storage at the same time. The system can be divided into segments following the hierarchy. Segments at the same level may then be overlayed in core. This is probably unnecessary on a computer system which has virtual storage.

The segmentation is not achieved by using CALL OVERLAY instructions in the program code, but by loader control cards. On a CDC system the control cards define a segmented load. On an IBM system the control cards define an overlay program.

Figure 13-1 shows the tree structure defined by the CDC control cards, used to construct a segmented load, shown in Figures 13-2. The tree structure contains the main module located in each segment but other modules at the same level in the hierarchy may be included with the main module, in order to reduce the number of segments. For example module MODBAN calls modules RSE2 and OSE2. The tree structure shows only module RSE2 at the level below MODBAN. Module OSE2 is included in the same segment as RSE2.

As shown in the control cards in Figure 13-2, modules are included in a segment via INCLUDE commands. The control cards do not include all modules since the CDC loader usually includes modules at their lowest possible level consistent with the CALLs within the program code. Examination of the load map produced by any loader will show the location of modules and their references (CALLs) so that the control cards can be modified to optimize the load.

Some modules have been duplicated in several segments via the INCLUDE cards. These modules are not required by all modules at the level at which they are included and are duplicated instead of using more core to include them at a higher level.

The core storage required by the system depends on the dimensions specified (see paragraph 3.2 and Appendix A) but probably the largest core storage is required when the transit assignment segments are executed. The difference in core for these segments and other segments is relatively small however as would be expected with a reasonably designed segmented load.

As shown on the control cards certain COMMON areas are declared GLOBAL. There areas must be in core at all times. COMMON STRUC is declared GLOBAL to segment TRANSIT in which it is located. This declaration implies that modules below TRANSIT in the tree structure use the COMMON in TRANSIT. Other COMMON areas are duplicated in (local to) several segments as shown in the table in Figure 13-3. On some systems this may not be allowed in which case these COMMON areas may be declared as the equivalent of GLOBAL, i.e. in core at all times, and more core storage will be required.

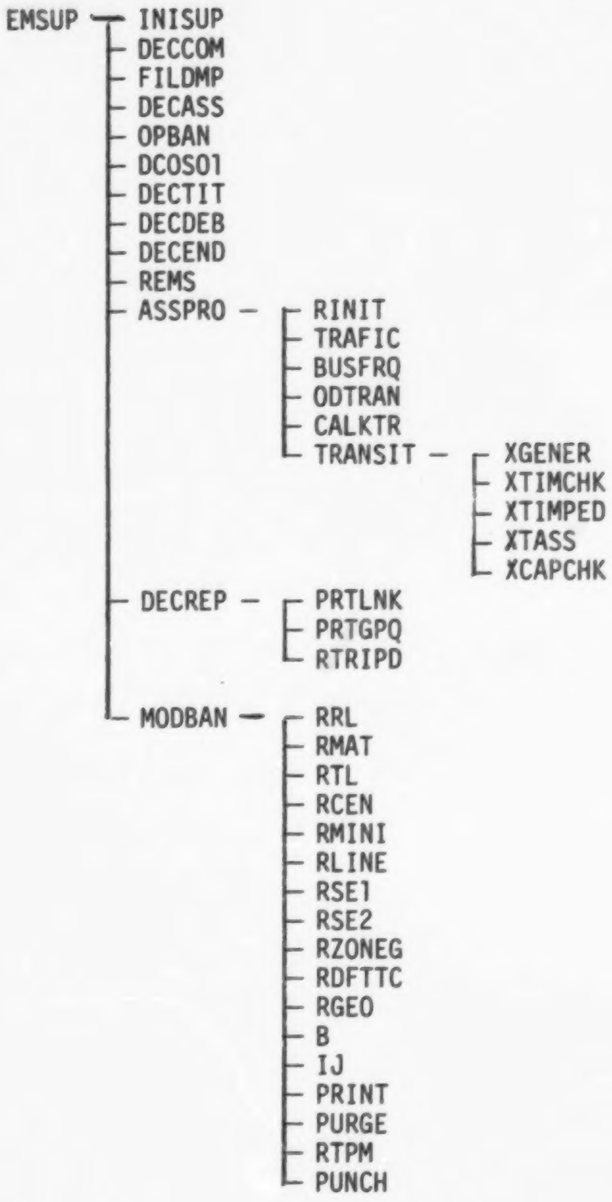

EMSUP
INISUP
DECCOM
FILDMP
DECASS
OPBAN
DCOSO1
DECTIT
DECDEB
DECEND
REMS
ASSPRO
RINIT
TRAFIC
BUSFRQ
ODTRAN
CALKTR
TRANSIT
XGENER
XTIMCHK
XTIMPED
XTASS
XCAPCHK
DECREP
PRTLNK
PRTGPQ
RTRIPD
MODBAN
RRL
RMAT
RTL
RCEN
RMINI
RLINE
RSE1
RSE2
RZONEG
RDFTTC
RGEO
B
IJ
PRINT
PURGE
RTPM
PUNCH

Figure 13-1 : Tree structure for segmented load

```
          GLOBAL  SUPCOM,CARTE
          GLOBAL  ABC,SE,WC,CHRSET,BG
          GLOBAL  FETBAN.COMBAN,COMEXT,PRGFLO,COMTPM,OH,AUXSE
          GLOBAL  MEM,RLCOM
          GLOBAL  APARM,NETSIZ,UNIT,REP,CONST
          GLOBAL  IO.BUF.,STP.END,FCL.C.,Q8.IO.,CON.RM,AOB.RM,JMPS.RM
          GLOBAL  MEMC.RM.OPES.FO.OPEN.FO,PUT.RT,CLSF.FO,TERM.RM
          GLOBAL  GET.FO,GET.BT,GET.RT,PUT.FO,CLSV.FO
          GLOBAL  REW.FO,SKFL.FO,RPAR.XX
          GLOBAL  MEM,RLCOM
 MAIN     TREE    EMSUP-(INISUP,DECCOM,FILDMP,XMODBAN.DECASS,OPBAN,DCOS01,
,XASSPRO,XDECREP.DECTIT,DECDEB.DECEND,REMS)
*
 XASSPRO TREE     ASSPRO-(RINIT,TRAFIC,TRANCOM,BUSFRQ,ODTRAN,CALKTR,PRTAS
,S)
 XDECREP TREE    DECREP-(PRTLNK,PRTGPQ,RTRIPD)
 TRANCOM TREE     TRANSIT-(XGENER,XTIMCHK,XTIMPED,XTASS,XCAPCHK)
*
 ASSPRO   INCLUDE PTNODE,UPACOD,PACKOD,GTNODE,PSLINK,RDLINK,RDRL
 ASSPRO   INCLUDE PSOD,RDOD,RDODM,RLINKU,RLINKV
 ASSPRO   INCLUDE PSDCOK,RDDCOK,GTLINE
 DECREP   INCLUDE PTNODE,UPACOD,GTNODE,PSLINK,RDLINK,RDRL
 DECREP   INCLUDE PSOD,RDOD,RDODM,RLINKU,RLINKV
 DECREP   INCLUDE ANWORD.SKACH.DECPAR,GTWORD
 FILDMP   INCLUDE PSDCOK,RDDCOK.GTLINE.GTNODE,PTNODE,PSOD,RDOD,RDODM
*
 DECASS   INCLUDE ANWORD,SKACH,DECPAR,GTWORD
 DECCOM   INCLUDE ANWORD,SKACH
 DECDEB   INCLUDE GTWORD,SKACH
 DECTIT   INCLUDE GTWORD,SKACH
 DECEND   INCLUDE GTWORD,SKACH
*
 TRAFIC   INCLUDE TRAFIC,BUSIN.NETWRK,CALCK,RASS,RASSAV
 TRAFIC   INCLUDE MINTRE
 TRAFIC   INCLUDE LOADTR,OBJFN,TLCST,TDCST,USECST,DEMCST,MOES,OPSTEP
 TRAFIC   INCLUDE LINKIN,FINDLC,FINDCC
 PRTLNK   INCLUDE NETWRK,LINKIN,FINDLC,FINDCC,RDRLTV,CORRD2,CORRD4
 RINIT    INCLUDE RINIT,AUTRIP,INITDM
 RTRIPD   INCLUDE RINIT,AUTRIP,INITDM
 FILDMP   INCLUDE DEMCST
*
```

Figure 13-2: CDC control cards to construct segmented load

```
TRANSIT   INCLUDE TIMOUT,RTNET,GTLINE
XGENER    INCLUDE XGENER,BLDNET,LISTLI,EXPNET,EQUIVA,EQUIVR,LINOUT
XGENER    INCLUDE SEARCH,WARNIN,TTIME,TDISCO
XGENER    INCLUDE RDLTV2,BUSROD,BUSCAR,BUSTIM
XTIMCHK   INCLUDE XTIMCHK,CHANGE
XTIMCHK   INCLUDE RDLTV2,BUSROD,BUSCAR,BUSTIM
XTIMPED   INCLUDE XTIMPED,FORLAX,REVLAX,ACCESX
XTIMPED   INCLUDE TPATH,FORLAB,REVLAB,LABEL,INSER,MODIF,ACCESS,DELETE
XTASS     INCLUDE XTASS,TDEM,LOASEC,FLOOUT,FORLAX,REVLAX,ACCESX
XTASS     INCLUDE TPATH,FORLAB,REVLAB,LABEL,INSER,MODIF,ACCESS,DELETE
XCAPCHK INCLUDE XCAPCHK
FILDMP  INCLUDE FILDMP
DECREP  INCLUDE DECREP
DECDEB  INCLUDE DECDEB
DECEND  INCLUDE DECEND
DECCOM  INCLUDE DECCOM
DECASS  INCLUDE DECASS
DECTIT  INCLUDE DECTIT
ASSPRO  INCLUDE ASSPRO
PRTLNK  INCLUDE PRTLNK
RTRIPD  INCLUDE RTRIPD
TRANSIT INCLUDE TRANSIT
*
 TRANSIT GLOBAL  STRUC
*
 XMODBAN TREE    MODBAN-(RRL,RMAT,RTL,RCEN,RMINI,RLINE,RSE1,RSE2,RZONEG
,,RDFTTC,RGEO,B,IJ,PRINT,RTPM,PUNCH)
 MODBAN    INCLUDE MODBAN,KEEPND,KPND1
 B         INCLUDE B,BCEN,BMINI,BRL,BTRO,BLINE,BTL,BMAT,BSE,BZONEG,BDFTTC
,,BGEO,BTPM,BTL1,BMAT1,BMAT2
 IJ        INCLUDE IJ,TCEN,IMINI,SRL,ITRO,ILINE,ITL,IMAT,ISE,IDFTTC,
,IGEO,ITPM,ITL2,IMAT1,IMAT2
 IJ        INCLUDE BSORTF,SRL0,SRL1,SRL2,ISE1
 RRL       INCLUDE RRL,ORL
 RMAT      INCLUDE RMAT,OMAT
 RTL       INCLUDE RTL,OTL
 RCEN      INCLUDE RCEN,OCEN
 RMINI     INCLUDE RMINI,OMINI
 RLINE     INCLUDE RLINE,OLINE
 RSE1      INCLUDE RSE1,ASE,MSE,OSE1
 RSE2      INCLUDE RSE2,ASE,MSE,OSE2,ASE2,MSE2
 RZONEG    INCLUDE RZONEG,OZONEG
 RDFTTC    INCLUDE RDFTTC,ODFTTC
 RGEO      INCLUDE RGEO,OGEO
 RTPM      INCLUDE RTPM,OTPM
 B         INCLUDE B,PURGE,BGEO,BSE
 REMS      INCLUDE REMS,IMAT,IMAT1,KEEPND,KPND1,ITL
 REMS      INCLUDE B,BCEN,BMINI,BRL,BTRO,BLINE,BTL,BMAT,BSE,BZONEG,BDFTTC
,,BGEO,BTPM,BTL1,BMAT1,BMAT2
 REMS      INCLUDE IJ,TCEN,IMINI,SRL,ITRO,ILINE,ITL,IMAT,ISE,IDFTTC,
,IGEO,ITPM,ITL2,IMAT1,IMAT2
 REMS      INCLUDE BSORTF,SRL0,SRL1,SRL2,ISE1
          END      EMSUP
```

Figure 13-2: CDC control cards to construct segmented load (continued)

Segment	Local COMMON areas
FILDMP	RNET1
	RNET2
	RVAR1
	RVAR2
	REXOD
	IX
	EXPA
DCOS01	IX
RINIT	REXOD
	IX
TRAFIC	RNET1
	RNET2
	RVAR1
	RVAR2
	REXOD
	RBUS
	RSEC
	WKMEM
	RUSER (Defined by sample user code)
GENER	EXPA
XTIMCHK	EXPA
XTIMPED	TRPATH
	HEAP
	CSX
XTASS	TRPATH
	HEAP
	CSX
	LOADT
	TRWALK
PRTLNK	RNET1
	RNET2
	RBUS
	WKMEM
	RUSER (Defined by sample user code)
RTRIPD	REXOD
	IX

Figure 13-3 : Table of local COMMON areas

14. User supplied routines

Certain functions implied by EMME must be defined and calibrated by the user for the city to which EMME will be applied. They are introduced into the EMME system installed by the user via the subprograms described in this section.

Subprogram	Purpose
TURLIN	Determine if road link is a turning link
LINKIN	Set up data required for road link delay function characteristics
FINDLC	Pick up road link characteristics
BUSROD	Determine bus time as a function of car time.

The EMME system is supplied with sample versions of these subprograms which apply to the sample problem. They should be replaced by the user's version.

14.1. User supplied routine: Function TURLIN

Logical function TURLIN determines if a link type specifies a turning link or not. The function is set to TRUE if it does, FALSE otherwise. Figure 14-1 shows the code for the sample function supplied with the EMME system, applicable to the sample problem. LTYPE is the road link type as specified on the road link coding form (see EMME User Guide)... For the sample problem, turning links are link types 90 through 99.

TURLIN is used by the data bank routines to determine which nodes are turning nodes (at each end of turning links).

TURLIN is also used by road traffic routine 6.68 MOES and transit routine 5.14 EXPNET.

14.2. User supplied routine: Subroutine LINKIN and FINDLC

Subroutine LINKIN allows the user to read data from the data bank or from any file he defines or to define values which are necessary to retrieve the data required by subroutine FINDLC. The data can be transmitted from LINKIN to FINDLC by a COMMON area defined by the user. Subroutine FINDLC determines the link characteristics necessary to calculate the travel time on a road link (by car).

The road traffic assignment routines 6.71 USECST and 6.72 TLCST use the BPR curve to determine travel time on a link as a function of the volume on the link. The form of this curve is:

$$t = len * \frac{60}{spd} \left(1 + a \left(\frac{vol}{cap}\right)^b \right)$$

where

t = travel time (minutes)

spd = free flow speed, no congestion (mph)

vol = volume on link (vehicles per hour)

cap = practical capacity of link (vehicles per hour)

$\begin{matrix} a \\ b \end{matrix}$ = coefficients determined by calibration

len = length of the link (miles)

Normally road links are classified according to a link type which is specified on the road link coding form.

The arguments of subroutines LINKIN and FINDLC are defined as follows:

Arguments	Type	Definition
NLINK	Integer	Input to LINKIN. Number of links in road network.
L	Integer	Input to FINDLC. Link number of link for which characteristics required.
LTYPE	Integer	Input to FINDLC. Link type of link for which characteristics required.
SPDFF	Real	Output from FINDLC. Free flow speed for the link L.
CAP	Real	Output from FINDLC. Total practical capacity for the link L.
RNULA	Real	Output from FINDLC. Number of lanes on link L.
A	Real	Output from FINDLC. Coefficient a of BPR curve described above.
B	Real	Output from FINDLC. Coefficient b of BPR curve as described above.

Note that if A is zero, congestion has no effect.

Figures 14-2 and 14-3 show the code for the sample subroutines supplied with the EMME system, applicable to the sample problem. For the sample problem there are 3 link types 1, 2 and 98. The coefficients a, b are assumed to be calibrated for each link type. Capacity per lane and free flow speed are assumed to be the same for all links of a given type. The values are as follows:

Link type	a	b	free flow speed	capacity per lane
1	0.0	1.0	10.0	700
2	.15	4.0	17.5	250
98	0.0	1.0	60.0	100

These values are defined in subroutine FINDCC (find curve characteristics) called by FINDLC. The capacity of a link is calculated from the number of lanes and the capacity per lane in subroutine FINDLC. The sample problem data contains the number of lanes for each link as the first item of user supplied data on the road link coding form. This data is read in using subroutine RLINKU (read user defined data; described for file 211 RLINK) in subroutine LINKIN and the values are stored in COMMON /USER/ a user defined COMMON area which transmits this data to FINDLC.

14.3. User supplied routine: Function BUSROD

Provision has been made in EMME to allow the user to specify his own model relating transit vehicle time to private car time on road links. This is achieved by replacing FUNCTION BUSROD (module 4.43) supplied with EMME system with the appropriate user-defined routine.

The version of BUSROD, which is described here was developed for the networks of the city of Winnipeg and it should only be considered as an example.

Transit link times are initialised by module 5.15 TTIME and are updated by module 4.31 TIMCHK. In all cases where the transit link is also a road link, function BUSROD is called in order to calculate the link time. Note that in such cases user-defined transit-only link times are ignored.

The parameters of BUSROD are the following:

Variable	Type	Description
LINK	Integer	address of the road link in arrays DIST and TIME.
TSPEED	Real	mean line speed (mph).
LITYP	Integer	line type

Two arrays, TIME and DIST are transmitted through common /EXPA/. TIME is the private car time on the road link (minutes) and DIST is the length of the link (miles). When defining BUSROD, special attention should be paid to the dimensions of the variables in common /EXPA/. They should be the same as those in the version of EMME installed by the user and not as in the following sample code.

For each line, the type stored in the data base is a two-digit decimal integer[1]. The rightmost digit, which specifies the model to be used for the determination of transit times, is the one transmitted through parameter LITYP.

The sample code which follows (Figures 14-4, 14-5) specifies the following model:

LITYP=1	bus time = car time
LITYP=2	bus time by mile = car time by mile + 2.15
LITYP=3	calculate time from bus speed and road link length
LITYP=other	bus time by mile = car time by mile + 2.1 * SQRT (car time by mile)

Any other line type classification may be used in order to specify the model, provided it has been clearly defined for the coding of transit line data.

Special attention should be paid to the segmentation of the EMME system if the user-defined code introduces new subprograms (see section 13).

---

[1] See section on transit network coding in User Guide.

```
      LOGICAL FUNCTION TURLIN (LTYPE)
*
*--------USER FUNCTION TO DETERMINE IF LTYPE SPECIFIES A TURNING LINK
*
*--------"EXAMPLE" VERSION
*
      TURLIN = .FALSE.
      IF (LTYPE .GE. 90) TURLIN = .TRUE.
      RETURN
      END
```

Figure 14-1

```
      SUBROUTINE LINKIN (NLINK)
*
*--------USER SUBROUTINE TO READ USER LINK DATA - "EXAMPLE" VERSION
*
      COMMON /RUSER/ NULA(250)
      REAL NULA
      LOGICAL EOFIL
*
      L = 0
   55 CALL RLINKU(UDATA,1,EOFIL)
      IF (EOFIL) GO TO 200
*
      L = L + 1
      NULA(L) = UDATA
      IF (L .LT. NLINK) GO TO 55
*
  200 RETURN
      END
```

Figure 14-2

```
      SUBROUTINE FINDLC (L,LTYPE,SPDFF,CAP,RNULA,A,B)
*
*--------USER WRITTEN SUBROUTINE TO RETRIEVE LINK CHARACTERISTICS
*         NECESSARY FOR DELAY CURVE IN BPR FORM
*
      COMMON /RUSER/ NULA(250)
      REAL NULA
*
*--------FIND CURVE CHARACTERISTICS CORRESPONDING TO LINK TYPE
      CALL FINDCC (LTYPE,SPDFF,CAPPL,A,B)
*
      CAP = CAPPL * NULA(L)
      RNULA = NULA(L)
*
      RETURN
      END
```

```
      SUBROUTINE FINDCC (LTYPE,SPDFF,CAP,A,B)
*
*--------USER WRITTEN SUBROUTINE TO DETERMINE PARAMETERS FOR
*         BPR DELAY CURVE
*
*
*--------"EXAMPLE" VERSION
*
*         THE LINK TYPE SPECIFIES THE SET OF CHARACTERISTICS TO USE
*         FOR A LINK
*         CAPPL = CAPACITY PER LANE
*
      DIMENSION SPD(3), CAPPL(3), ALPHA(3), BETA(3)
      DATA SPD/ 10.0, 17.5, 60.0/
      DATA CAPPL/ 700., 250., 100. /
      DATA ALPHA/  0.0,  .15,  0.0 /
      DATA BETA /  1.0,  4.0,  1.0 /
*
      LT = LTYPE
      IF (LTYPE .GT. 2) LT = 3
      SPDFF = SPD(LT)
      CAP = CAPPL(LT)
      A = ALPHA(LT)
      B = BETA(LT)
      RETURN
      END
```

Figure 14-3

```
      FUNCTION BUSROD(LINK,TSPEED,LITYP)
*
*        MODULE  4.43  B U S R O D
*        PURPOSE: DETERMINE TRANSIT VEHICLE TIME ON A ROAD LINK
*                  - IF A TIME VALUE FOR PRIVATE CARS IS DEFINED
*                    IN ARRAY  TIME  FOR THE CORRESPONDING LINK
*                    THE TRANSIT LINK TIME IS CALCULATED BY CALLING
*                    FUNCTION  BUSCAR
*                  - OTHERWISE, THE TRANSIT VEHICLE TIME IS CALCULATED
*                    FROM TRANSIT SPEED  TSPEED  AND LINK LENGTH  DIST
*
*        BUSROD IS CALLED BY MODULE 5.15 TTIME AND 4.31 TIMCHK
*
*
*        THIS IS THE BUS TIME - CAR TIME MODEL
*        TO BE USED ON THE WINNIPEG NETWORK
*
*
*
      COMMON /EXPA/ CORNOD(400),TURNIN(3000),DIST(3000),
     +              TIME(3000),
     +              NOT(200),NOA(200),LINKA(200),NOR(200),LINKR(200),
     +              NOLAB(951),NDEST(2400)
      INTEGER CORNOD
      LOGICAL TURNIN
      INTEGER NOT,NOA,LINKA,NOR,LINKR,NOLAB,NDEST
      INTEGER NA,NR,NT
      EQUIVALENCE (LINKA(1),NA),(LINKR(1),NR)
      T=TIME(LINK)
      D=DIST(LINK)
      CARTIM=0.0
      IF(D .LE. 0.0) GO TO 25
      IF(T.LT.0.0) GO TO 10
*
*- TEST FOR EXPRESS SERVICE
*
      IF(LITYP.NE.   1) GO TO 5
*-- BUS TIME = CAR TIME FOR EXPRESS
      CARTIM=T
      GO TO 20
    5 CONTINUE
*
*- TEST FOR CONSTANT SPEED BUS
*
      IF(LITYP .EQ.   3) GO TO 10
*- DEFINE CAR TIME BY MILE
      T=T/D
      CARTIM=BUSCAR(T,LITYP)
      CARTIM=CARTIM*D
      GO TO 20
   10 CARTIM=D/TSPEED *60.0
   20 IF(CARTIM.LT.0.0) CARTIM=0.0
   25 BUSROD=CARTIM
      RETURN
      END
```

Figure 14-4

```
    FUNCTION BUSCAR(CTIME,LTYPE)
*
*     MODULE  4.42  B U S C A R
*     PURPOSE: CALCULATE BUS TIME BY MILE (BUSCAR)
*              GIVEN CAR TIME BY MILE (CTIME)
*              AND LINE TYPE (LTYPE)
*
*
*
*     THIS IS THE BUS TIME - CAR TIME MODEL
*     TO BE USED ON THE WINNIPEG NETWORK
*
*
    IF(LTYPE .EQ.  1) GO TO 20
    IF(LTYPE .EQ.  2) GO TO 10
*
*- REGULAR BUS
*
    BUSCAR=CTIME + 2.1*SQRT(CTIME)
    RETURN
*
*- FAST BUS
*
  10 BUSCAR=CTIME + 2.15
    RETURN
*
*- EXPRESS SERVICE
*
  20 BUSCAR=CTIME
    RETURN
    END
```

Figure 14-5

Volume

2

UNIVERSITÉ DE MONTRÉAL

CENTRE DE RECHERCHE SUR LES TRANSPORTS

CASE POSTALE 6128, STATION A, MONTRÉAL, QUÉBEC, H3C 3J7

EMME

SYSTEMS PROGRAMMERS GUIDE

VOLUME II

by

Linda James-Lefebvre, Claude Achim,
Caroline Fisk and Jacques Lefebvre

This work was supported by a grant from the Research and Development Centre of Transport Canada

Université de Montréal
Centre de recherche sur les transports - Publication # 124
Janvier 1979

Dépôt légal - 1er trimestre 1979

TABLE OF APPENDICES

A: COMMON area descriptions 98
B: Module specifications. 153
C: File descriptions. 491
D: Cross reference module and file index. 563
E: Variable demand traffic assignment algorithm 586

APPENDICES

Appendix A: COMMON area descriptions

This appendix contains descriptions of the most important COMMON areas. The descriptions are arranged in alphabetical order by COMMON area name. Each description shows the Fortran code defining the area and may contain symbols for the variable dimensions used in coding the EMME system. The symbols begin and end with a percentage sign and are replaced by an appropriate value when an EMME system is generated for a particular user or application (see also paragraph 3.2).

Following is an explanation of the significance of the symbols:

%CENT% = maximum number of centroids
%CENT+1% = %CENT%+1
%RNODE+1% = maximum number of road nodes (turning + regular) + 1
%NODE% = maximum of road nodes + centroids
%NODE+1% = %NODE%+1
%NODET% = maximum number of nodes that can be used in the coding of the transit network (all regular road nodes + all transit-only nodes)
%NODES% = maximum number of nodes on the bimodal network including centroids
%LINK% = maximum number of road links
%LINK2% = maximum number of road links between nodes of the road network (i.e. links connecting the centroids to the network are not included)
%LINK2+T% = maximum length of a list containing all road links between nodes of the road network plus all the user-defined transit-only links
%3LINK2% = 3 times %LINK2%
%LINE% = maximum number of transit lines
%SEG% = maximum length of the list containing all the line itineraries
%SEL% = maximum length of the list of transit lines passing at each node
%SEGL% = maximum number of nodes on the itinerary of a transit line

%TACCES% = maximum number of transit access links
= maximum number of transit egress links

%CLIST% = maximum length of the node expansion list

%LINDIV% = maximum number of lines included in a set of common lines for the line diversion algorithm in transit path calculations

%LINKU% = maximum number of lines passing at a single node

%*2% = *2 for an IBM system if half word (2 byte) integer variables are allowed
= blank for a CDC system where all integer variables are full word.

The following symbols are used in the data bank section only:

%L.BLOC% = number of words in a block of the data bank

%PRUBL% = number of physical record units (PRU) per block

%NCPW% = number of characters per word

%MAXMEM% = maximum size of array MEM used for dynamic allocation of memory.

COMMON / APARM /

```
 COMMON /APARM/ ASSPSW(15),
+              LINDIV,USEWP,
+              BVOLSW,VALID,USEPLT,EPSTIM,WFACT,TOLERZ,
+              WPENAL,WAITMI,WAITMA,WWALK,WWAIT,EPSIR,EPSIL,
+              HDWSET(20),MAXHDW,TVCAP,
+              TMAX,TDSPED,TDTIME,SPLT,AOCC,BUSFP,
+              SPLTV,AOCCV,BUSFAC,ALFEXP,VADLIM,
+              IDEN(10),ITRD,
+              MXRD,MXLT,MXCP,DFNC
 EQUIVALENCE (ASSPSW(1),TNETSW), (ASSPSW(2),BUSFSW),
+             (ASSPSW(3),TIMPSW), (ASSPSW(4),ROADSW),
+             (ASSPSW(5),VELTSW), (ASSPSW(6),TRANSW),
+             (ASSPSW(7),VECPSW), (ASSPSW(8),TVARSW),
+             (ASSPSW(9),BIMOSW),(ASSPSW(10),RVARSW),
+             (ASSPSW(11),BUSISW),(ASSPSW(12),ODEXSW)
 LOGICAL ASSPSW, TNETSW,BUSFSW,TIMPSW,ROADSW,VELTSW,TRANSW,
+        VECPSW,TVARSW,  BIMOSW,RVARSW,USEPLT,VALID,BVOLSW,
+        BUSISW,LINDIV,USEWP,ODEXSW
 INTEGER SPLT,AOCC,TVCAP,
+        BUSFP
 INTEGER%*2% MXRD,MXLT,MXCP,DFNC
```

Description of variables

/APARM/variables contain the requirements for the current assignment.

Variable	Type	Description
TNETSW	LOGICAL	TRUE indicates generation of a transit network required
BUSFSW	LOGICAL	TRUE indicates bus frequency on road links required
TIMPSW	LOGICAL	TRUE indicates transit impedance computed
ROADSW	LOGICAL	TRUE indicates road traffic assignment required
VELTSW	LOGICAL	TRUE indicates automatic modification of transit link times based on predicted road link times is required
TRANSW	LOGICAL	TRUE indicates transit assignment required
VECPSW	LOGICAL	TRUE indicates automatic modification of transit capacity, based on predicted demand, is required
TVARSW	LOGICAL	TRUE indicates variable demand transit assignment required

Variable	Type	Description
BIMOSW	LOGICAL	TRUE indicates bimodal assignment is required
RVARSW	LOGICAL	TRUE indicates variable demand road traffic assignment required
BUSISW	LOGICAL	TRUE indicates bus frequency on road links (observed or computed) required
ODEXSW	LOGICAL	TRUE indicates a vehicle adjustment matrix is to be used
LINDIV	LOGICAL	TRUE indicates transit assignment will use line diversion algorithm
USEWP	LOGICAL	TRUE indicates transit path multipliers from databank to be used
BVOLSW	LOGICAL	TRUE indicates that bus volume on road links is included in road traffic assignment
VALID	LOGICAL	TRUE indicates transit network validation is required
USEPLT	LOGICAL	TRUE indicates initial transit times will be based on predicted road link times. FALSE indicates observed road link times.
EPSTIM	REAL	Significant relative difference when comparing transit link times before and after modification (see parameter TOLT)
WFACT	REAL	Factor applied to transit headways to determine transit waiting time
TOLERZ	REAL	Significant relative difference when comparing transit impedances (see parameter TOLU)
WPENAL	REAL	Transit waiting penalty (see parameter WPEN)
WAITMI	REAL	Minimum transit waiting time

Variable	Type	Description
WAITMA	REAL	Maximum transit waiting time
WWALK	REAL	Global value for transit path multiplier for walk time
WWAIT	REAL	Global value for transit path multiplier for wait time
EPSIR	REAL	Significant relative change in objective function of road traffic assignment
EPSIL	REAL	Significant relative difference when comparing path times in road traffic assignment
HDWSET	REAL	Permissible transit headways for capacity adjustment (minutes)
MAXHDW	INTEGER	Number of values in HDWSET
TVCAP	INTEGER	Transit vehicle capacity
TMAX	REAL	Maximum CPU seconds allowed for each road traffic assignment
TDSPED	REAL	Default transit speed (see parameter DSPD)
TDTIME	REAL	Default transit travel time (see parameter DTIM)
SPLT	INTEGER	Indicates modal split specification to be used (4 characters)
AOCC	INTEGER	Indicates auto occupancy specification to be used (4 characters)
BUSFP	INTEGER	Indicates bus frequency specification to be used (4 characters)
SPLTV	REAL	Global value for modal split
AOCCV	REAL	Global value for auto occupancy
BUSFAC	REAL	Automobile equivalent of one bus (automobiles)

Variable	Type	Description
ALFEXP	REAL	Express bus time factor (see parameter EXPF)
VADLIM	REAL	Lower limit on demand between an OD pair for that demand to be allowd to vary
IDEN	INTEGER	Contains 40 characters from IDENT subcommand (4 characters per word)
ITRD	INTEGER	Current number of iterations in a road traffic assignment (equivalent to NUMCYC in 6.60 RASS)
MXRD	INTEGER	Maximum number of iterations allowed in each road traffic assignment
MXLT	INTEGER	Maximum number of times transit link times may be modified (see parameter ITLT)
MXCP	INTEGER	Maximum number of times transit capacity may be modified (see parameter ITCP)
DFNC	INTEGER	Indicates demand function type to be used (1 = logit, 2 = linear)

Variables TNETSW through ODEXSW are equivalenced to the vector ASSPSW (module 4.00 ASSPRO switches but also used in other modules).

Use

The variables in APARM provide the interface between the options and parameter, specified by the user in the $ASSIGN command and its subcommands,and the EMME assignment routines. Details of the correspondence between the options and parameters and the /APARM/ variables can be found in the descriptions of modules 1.60 DECASSS and 1.62 DECPAR.

In general, LOGICAL variables are used to determine if a routine is executed or not, or which of two routines to execute. INTEGER and REAL variables contain values, used in the assignment routines, which may be chosen by the user.

With the following exceptions, no variables in /APARM/ are changed by the assignment routines:

ITRD	is used in the road traffic assignment (module 6.60 RASS).
EPSIR	is set at the end of each road traffic assignment (module 6.60 RASS) to the final relative change in the objective function.
BVOLSW	is used to indicate that bus volume on road links is available and should be picked up (module 6.60 RASS) for use in calculating link times (modules 6.71 USECST and 6.72 TLCST). This variable is used so that if the bus volume on road links feature is never used, the vector BUSVOL may be defined as having only 1 element.

Initialisation

All the variables in /APARM/ except EPSIR, ITRD and BVOLSW are set to their default values by module 1.60 DECASS which is called when a $ASSIGN command is found. The options on the $ASSIGN command are decoded and set variables as described in module 1.60 DECASS and also some default values as described in module 1.62 DECPAR. Any subcommands following the $ASSIGN command are decoded as described is modules 1.60 DECASS abd 1.62 DECPAR, setting values in the variables in /APARM/.

Variable EPSIR is initialised in module 6.60 RASS and cannot be changed by a subcommand.

Variable ITRD, equivalent to NUMCYC is set to 1 in 6.60 RASS.
Variable BVOLSW is set to FALSE in 6.20 TRAFIC.

References

1.00	EMSUP
1.60	DECASS
1.62	DECPAR
4.00	ASSPRO
4.21	TIMPED
4.31	TIMCHK
4.41	BUSTIM
4.61	CAPCHK
5.10	GENER
5.13	TIMOUT
5.14	EXPNET
5.15	TTIME
5.19	WARNIN
5.22	FORLAB
5.23	REVLAB
5.32	FORLAX
5.33	REVLAX
5.41	ODTRAN
5.50	TASS
5.55	TDEM
5.56	LOASEC
5.60	FLOOUT
5.90	CLAPRT
5.95	GPQDMP

6.00	RINIT
6.10	AUTRIP
6.15	INITDM
6.20	TRAFIC
6.25	CALCK
6.50	BUSIN
6.60	RASS
6.61	LOADTR
6.63	OBJFN
6.64	OPSTEP
6.73	DEMCST
6.74	TDCST
6.80	RASSAV
6.92	PRTGPQ
6.94	PRTLNK
6.96	RTRIPD

COMMON /CARTE/

```
COMMON /CARTE/ IMP,PRINTD,COL(80),EOFINP
LOGICAL IMP,PRINTD,EOFINP
INTEGER COL
```

Description of variables

Variable	Type	Description
IMP	LOGICAL	TRUE indicates contents of COL(80) to be printed.
PRINTD	LOGICAL	TRUE indicates contents of COL(80) have been printed
COL	INTEGER	Contents of last record read from COMFIL, 1 character per word left justified, blank filled.
EOFINP	LOGICAL	TRUE indicates EOF has been encountered when reading COMFIL.

Use

IMP	is used to control printing of command file records
PRINTD	is used to prevent printing a command file record more than once
COL	is analyzed by the decoding routines in the supervisor or data bank input routines
EOFINP	is set to TRUE by 1.11 RDCOM when EOF found in COMFIL

Initialization

IMP = FALSE	Set by 1.02 MODBAN according to LIST option on CORRECTION or USE subcommands
PRINTD = FALSE	Set by 1.11 RDCOM after a record has been successfully read from COMFIL.
COL	Filled by 1.11 RDCOM
EOFINP = FALSE	Set by 1.10 INISUP

References

0.09	PRT
0.11	SKBLK
0.12	SKSEP
0.13	SKNSEP
0.14	SKNBLK
1.01	ANCOM
1.02	MODBAN
1.09	BLOCK DATA
1.10	INISUP
1.11	RDCOM
1.12	ANWORD
1.13	SKACH
1.15	DECDEB
1.16	DECEND
1.30	DECCOM
1.40	DECTIT
1.60	DECASS
1.62	DECPAR
1.63	GTWORD
1.70	DECREP
2.42	RCEN
2.43	RMINI
2.44	RRL
2.45	RLINE
2.46	RTL

2.47	RMAT
2.63	RZONEG
2.64	RDFTTC
2.72	RSE1
2.76	AKEY
2.77	DKEY
2.78	RSE2
2.87	OCEN
2.88	OMINI
2.89	OLINE
2.90	ORL
2.91	OTL
2.92	OMAT
2.93	OZONEG
2.94	OSE1
2.95	OSE2
2.96	ODFTTC
2.98	SCENO
2.99	ACCUM
2.104	RGEO
2.105	OGEO
2.111	SCALE
2.112	RTPM
2.118	OTPM
3.40	MUCCA
3.41	SCEN1
3.53	PSCEN
3.69	USCEN

COMMON / CONST /

```
COMMON /CONST/ INFINI,XINFIN,NOTLAB,PERLAB
INTEGER PERLAB
```

Description of variables:

Variable	Type	Description
INFINI	INTEGER	large integer number used whenever an infinite integer constant is needed
XINFIN	REAL	the real equivalent of INFINI
NOTLAB	INTEGER	constant used to indicate that a node is not labelled
PERLAB	INTEGER	constant used to indicate that a node has received a permanent label

Use:

NOTLAB and PERLAB are used in transit shortest paths calculations.

Initialisation:

BLOCK DATA

References:

1.00	EMSUP
2.104	RGEO
2.105	OGEO
2.110	IGEO
2.42	RCEN
2.87	OCEN
3.14	GTNODE
4.21	TIMPED
4.41	BUSTIM
5.02	DISTAN
5.15	TTIME
5.20	TPATH
5.22	FORLAB
5.23	REVLAB
5.24	LABEL
5.32	FORLAX
5.33	REVLAX
5.50	TASS
5.55	TDEM
5.60	FLOOUT
6.25	CALCK
6.60	RASS
6.61	LOADTR
6.63	OBJFN

8.09	BLOCK DATA
8.21	PLOTVN
8.22	DINAUN
8.41	PLOTVV
8.42	DINAUV
8.44	VOLPLT
8.45	AVALGO
8.47	AVLINE
8.61	PLOTTN
8.62	DINTRN
8.74	TNACES
8.81	PLOTTV
8.82	DINTRV
8.84	ORDER

COMMON / CSX /

```
COMMON /CSX/ LK(%LINEU%),NALK(%LINEU%),NK,NLIMIT,NLINX
```

Description of variables:

Variable	Type	Description
LK	INTEGER	list of lines passing at node IPIVOT to be scanned in the current step of the labelling process
NALK	INTEGER	address of node IPIVOT in array LISTN (common / STRUC /) for lines in LK
NK	INTEGER	number of lines in LK
NLIMIT	INTEGER	maximum number of common lines to be considered
NLINX	INTEGER	common lines are stored in LIAC (common / TRPATH /) modulo NLINX (see FORLAX, REVLAX)

Use:

To determine common line sections in FORLAX and REVLAX. It should be noted that lines in LK are stored in the same order as they appear in LISTL (common / STRUC /) and that the order is the same for any node.

Initialisation:

LK, NALK and NK are determined in module 5.31 ACCESX.

NLIMIT appears in DATA statement in module 5.31 ACCESX.

NLINX is set to NLINE + 1 in 5.31 ACCESX and 5.50 TASS.

Note: NLINX ** NLIMIT must fit in an INTEGER word.

References:

5.31	ACCESX
5.32	FORLAX
5.33	REVLAX
5.50	TASS
5.56	LOASEC

COMMON / EXPA /

```
 COMMON /EXPA/ CORNOD(%CLIST%),TURNIN(%LINK2+T%),DIST(%LINK2+T%),
+              TIME(%LINK2+T%),
+              NOT(%SEGL%),NOA(%SEGL%),LINKA(%SEGL%),NOR(%SEGL%),LINKR(%SEGL%),
+              NOLAB(%RNODE+1%),NDEST(%LINK2%)
 INTEGER CORNOD
 LOGICAL TURNIN
 INTEGER%*2% NOT,NOA,LINKA,NOR,LINKR,NOLAB,NDEST
 INTEGER%*2% NA,NR,NT
 EQUIVALENCE (LINKA(1),NA),(LINKR(1),NR)
```

Description of variables:

Variable	Type	Description
CORNOD	INTEGER	list of road nodes retrieved from the correspondence table (file NODCOR)
TURNIN	LOGICAL[+]	.TRUE. if the road link is an intersection crossing link (expanded intersections) .FALSE. if the link is a regular road link
DIST	REAL[+]	length of the link
TIME	REAL	travel time on the link
NOT	INTEGER	list of nodes to be removed from the transit line itinerary in order to establish correspondence with road network
NOA	INTEGER[++]	sequence of nodes describing the forward itinerary of the transit vehicle on the road network
LINKA	INTEGER[++]	sequence of road links corresponding to NOA
NOR	INTEGER[++]	sequence of nodes describing the reverse itinerary of the transit vehicle on the road network
LINKR	INTEGER[++]	sequence of road links corresponding to NOR
NOLAB	INTEGER[++]	list of road links between nodes of the road network
NDEST	INTEGER[++]	list of road links between nodes of the road network

---

[+] The bottom part of TURNIN and DIST is used to store the user defined transit-only links (see below)

[++] Two byte words in IBM version

Use:

Arrays NOT, NOA, LINKA, NOR, LINKR which are saved on file CORNET specify the itinerary of transit vehicles on the road network. They are used to determine bus volumes on road links and to calculate transit travel times from road link times.

Initialisation:

CORNOD = defined from file NODCOR in 5.14 EXPNET

TURNIN
DIST
TIME
= 1) top part (1 to number of links between road nodes) read from file RLINK in 5.14 EXPNET

2) bottom part (after road links)
module 5.15 TTIME contains the following
EQUIVALENCE (TRORIG(1),TURNIN(1)) (TRDEST(1),DIST(1))

TRORIG will contain the origins of transit-only links
TRDEST will contain the destinations of transit-only links
TIME will contain the user-defined times

Transit-only links are read from file TLTIM

NOLAB
NDEST
= defined while reading road links in 5.14 EXPNET

NOT
NOA
LINKA
NOR
LINKR
= defined for each transit line by modules 5.14 EXPNET, 5.16 EQUIVA, 5.17 EQUIVR

References:

4.31	TIMCHK
4.41	BUSTIM
4.43	BUSROD
5.14	EXPNET
5.15	TTIME
5.16	EQUIVA
5.17	EQUIVR
5.18	SEARCH
5.92	PRCOR

COMMON / HEAP /

```
      COMMON /HEAP/ IPIVOT,NBHEAP,TSET(%NODET%),NSET(%NODET%),
     +              IDICO(%NODET%),HEAPAD
      INTEGER HEAPAD
      INTEGER%*2% NSET,IDICO
```

Description of variables:

Variable	Type	Description
IPIVOT	INTEGER	node from which line scanning takes place in the labelling process
NBHEAP	INTEGER	number of nodes in the heap
TSET	REAL	temporary time label of a node
NSET	INTEGER +	list of nodes in the heap
IDICO	INTEGER +	address of a node in the heap
HEAPAD	INTEGER	address in the heap of node to be labelled

Use:

In the labelling process of transit shortest path calculations, the nodes with temporary time labels are saved in a list having a heap structure. The node with the smallest time lable is always on the top of the heap.

Initialisation:

The heap is initialised by marking all nodes "not labelled" and then by labelling all access nodes with the corresponding walk time (modules 4.21 TIMPED, 5.50 XTASS). It is then updated by modules 5.25 DELETE, 5.26 INSER and 5.27 MODIF.

+ Two byte word on IBM version

References:

4.21	TIMPED
5.20	TPATH
5.21	ACCESS
5.22	FORLAB
5.23	REVLAB
5.24	LABEL
5.25	DELETE
5.26	INSER
5.27	MODIF
5.31	ACCESX
5.32	FORLAX
5.33	REVLAX
5.50	TASS

COMMON / LOADT /

```
 COMMON /LOADT/ DY(%SEG%),DZ(%SEG%),PASTOT(%LINE%),PASPAY(%LINE%),
+               ENTRE(%NODET%),SORT(%NODET%),UNBORD(%NODET%),
+               TWAIT,TVEH,DVEH
```

Description of variables:

Variable	Type	Description
DY	REAL	passenger volumes on forward itineraries of transit lines
DZ	REAL	passenger volumes on reverse itineraries of transit lines
PASTOT	REAL	total number of passengers boarding each transit line
PASPAY	REAL	total number of passengers paying fare on each transit line
ENTRE	REAL	number of passengers accessing at each node
SORT	REAL	number of passengers egressing at each node
UNBORD	REAL	number of passengers unboarding a transit vehicle at each node
TWAIT	REAL	total waiting time for an O-D pair
TVEH	REAL	total in-vehicle time for an O-D pair
DVEH	REAL	total in-vehicle distance for an O-D pair

Use:

To accumulate passenger volumes and collect transit assignment statistics in 5.56 LOASEC.

Initialisation:

All variables are set to zero in 5.50 TASS.

References:

5.50 TASS
5.56 LOASEC
5.60 FLOOUT

COMMON / NETSIZ /

```
      COMMON /NETSIZ/ NCENT,NTURN,NSPENO,NROAD,NNODE,
     +                NTRAN,NNODET,NNODES,
     +                NLINK,LINKS1,NLINKR,
     +                NLINE,NSEG,NSEL
```

Description of variables:

. NCENT thru NNODES = see Internal Node Numbers

. NLINK = number of road links

. LINKS1 = number of road links in section 1 of file RLINK

. NLINKR = number of road links between nodes of the road network

. NLINE = number of transit lines

. NSEG = number of line segments (length of the list containing line itineraries)

. NSEL = length of the list of transit lines passing at each node

Initialisation:

All the variables are initialised by module 3.21 OPBAN except NSEG and NSEL which are initialised by module 5.12 BLDNET.

References:

0.30	PACKOD
0.31	UPACOD
2.11	TCEN
2.117	ITPM
2.119	IJ
2.15	IMINI
2.16	SRLO
2.18	SRL
2.19	SRL1
2.23	ITRO
2.26	ILINE
2.32	ITL2
2.33	ITL1
2.41	IMAT2
2.60	KEEPND
2.82	ISE
2.85	ISE1
2.86	IDFTTC
3.11	RDODM
3.14	GTNODE
3.21	OPBAN
3.33	GTNODV
3.34	RDDVC
4.15	RDLTV2
4.18	BUSFRQ
4.21	TIMPED

4.31	TIMCHK
4.41	BUSTIM
4.61	CAPCHK
5.00	RTNET
5.01	WTNET
5.03	TDISCO
5.09	LINOUT
5.10	GENER
5.12	BLDNET
5.13	TIMOUT
5.14	EXPNET
5.15	TTIME
5.16	EQUIVA
5.17	EQUIVR
5.18	SEARCH
5.20	TPATH
5.30	CALKTR
5.31	ACCESX
5.41	ODTRAN
5.50	TASS
5.55	TDEM
5.56	LOASEC
5.60	FLOOUT
5.90	CLAPRT
5.92	PRCOR
5.94	PRIMPE
6.10	AUTRIP

6.15	INITDM
6.25	CLACK
6.41	NETWRK
6.50	BUSIN
6.60	RASS
6.61	LOADTR
6.62	MINTRE
6.63	OBJFN
6.64	OPSTEP
6.68	MOES
6.80	RASSAV
6.92	PRTGPQ
6.94	PRTLNK
6.95	RDRLTV
6.96	RTRIPD
7.10	DCOS01
8.21	PLOTVN
8.22	DINAUN
8.23	NETPLT
8.24	ANALGO
8.27	LINKPL
8.41	PLOTVV
8.42	DINAUV
8.43	TABLE
8.44	VOLPLT
8.45	AVALGO

8.47	AVLINE
8.48	CNUM
8.61	PLOTTN
8.62	DINTRN
8.63	ROUTE
8.64	TABTN
8.65	INSERT
8.68	TRPLOT
8.69	TNALGO
8.71	CHECTN
8.74	TNACES
8.81	PLOTTV
8.82	DINTRV
8.83	VOLACC
8.84	ORDER
8.85	ACC1
8.90	PLTRVL
8.91	TVALGO
8.93	CHECTV
8.95	TVACES
8.96	TVLINE
9.21	CORRD2

COMMON / RBUS /

```
COMMON /RBUS/ BUSVOL(%LINK%)
```

Description of variable

Variable	Type	Description
BUSVOL	REAL	BUSVOL(L) contains the automobile equivalent of the bus volume on link L, L=1,NLINK

NLINK is as described for COMMON /NETSIZ/

Use

Used only if BUSF=YES.

Read in from file TLFREQ for first traffic assignment. Saved on file RLTIVL and read from that file for any subsequent assignments.

Added to the automobile volume in order to calculate the travel time on a link.

Read from RLTIVL for reports.

Initialisation

Read from TLFREQ in module 6.50 BUSIN

Read from RLTIVL in module 6.60 RASS

References

6.50	BUSIN
6.60	RASS
6.63	OBJFN
6.64	OPSTEP
6.80	RASSAV
6.94	PRTLNK
6.95	RDRLTV

COMMON /REP/

```
COMMON /REP/  TITLE(60,2), TDAT, TITTIM, IPAGE
INTEGER TITLE
```

Description of variables

Variable	Type	Description
TITLE	INTEGER	Text printed in centre of second and third lines of titles, 60 characters each. Stored 1 character per word, left justified
TDAT	Undefined	Text for data (10 characters on CDC system)
TITTIM	Undefined	Text for time (10 characters on CDC system)
IPAGE	Integer	Printed on title line one.

Use

/REP/ contains the information printed in 3 title lines by module 0.01 TTIT.

TDAT and TITTIM are used in the identification record on all files produced by the assignment modules.

TITLE is filled, by module 1.40 DECTIT, with information from a $TITLE command.

IPAGE is set and incremented by routines calling TTIT.

Initialisation

TITLE = blanks	Set by BLOCK DATA
TDAT = current date	Set by call to a date subroutine in module 1.10 INISUP
TITTIM = current time	Set by call to a time subroutine in module 1.10 INISUP
IPAGE = 0	Set by module 1.10 INISUP

References

1.01	BLOCK DATA
1.10	INISUP
1.40	DECTIT
4.01	PRTASS
5.09	LINOUT
5.12	BLDNET
5.13	TIMOUT
5.41	ODTRAN
5.50	TASS
5.60	FLOOUT
5.95	GPQDMP
6.20	TRAFIC
6.60	RASS
6.80	RASSAV
6.92	PRTGPQ
6.94	PRTLNK
6.96	RTRIPD
8.02	PTITLE
8.21	PLOTVN
8.41	PLOTVV
8.43	TABLE
8.61	PLOTTN
8.81	PLOTTV
8.83	VOLACC
9.01	TTIT

COMMON / RNET1/

```
          COMMON /RNET1/ DIST(%LINK%),LTYP(%LINK%),
         +                NOLAB(%NODE+1%),ND(%LINK%)
          INTEGER%*2% LTYP,NOLAB,ND
```

Description of variables

/RNET1/ contains the specifications of the road network.

Variable	Type	Description
DIST	REAL	DIST(L) contains the length of link L in miles, L=1,NLINK
LTYP	INTEGER	LTYP(L) contains the link type, used to specify the delay curve number, of link L, L=1,NLINK
NOLAB	INTEGER	NOLAB(NORIG) contains the link number L of the first link which has link origin node or centroid number NORIG, NORIG=1,NNODE
ND	INTEGER	ND(L) contains the destination node number of link L, L=1,NLINK

NLINK and NNODE are described in COMMON/NETSIZ/.

Use

The variables in /RNET1/ are used as input to the road traffic assignment modules, the road traffic report modules and the graphics programs which plot the road network.

Initialisation

All the variables in /RNET1/ are filled in the module 6.41 NETWRK.

References

6.41	NETWRK
6.50	BUSIN
6.60	RASS
6.61	LOADTR
6.62	MINTRE
6.68	MOES
6.71	USECST
6.72	TLCST
6.80	RASSAV
6.94	PRTLNK
6.95	RDRLTV
8.21	PLOTVN
8.23	NETPLT
8.24	ANALGO
8.41	PLOTVV
8.43	TABLE
8.44	VOLPLT
8.45	AVALGO
8.47	AVLINE
9.21	CORRD2

COMMON / RNET2 /

```
COMMON /RNET2/ VCUM(%LINK%),TRAV(%LINK%),ODIJ(%CENT%),KD(%CENT%)
```

Description of variables

/RNET2/ is used to contain values used during a fixed or variable demand road traffic assignment

Variable	Type	Description
VCUM	REAL	VCUM(L) contains the automobile volume (vehicles per hour) predicted for link L, L=1,NLINK
TRAV	REAL	TRAVL(L) contains the automobile travel time (minutes) predicted per link L, L=1,NLINK
ODIJ	REAL	ODIJ(IQ) contains the number of automobile trips from the current origin to destination IQ, IQ=1,NCENT
KD	INTEGER	KD(IP) indicates non-zero (positive) or zero (negative) total demand from origin IP

NLINK and NCENT are as described in COMMON /NETSIZ/

Use

VCUM and TRAV contain the predicted link volumes and times at each road traffic assignment iteration . When this assignment is terminated they contain the predicted values for the assignment which are subsequently saved on file RLTIVL. VCUM and TRAV are also used to hold the predicted values when these are read in by module 6.95 RDRLTV for reports.

ODIJ is used during the road traffic assignment to hold the demand from the current origin, read from file GPQRAN. In a fixed demand assignment these

Values are not changed. In a variable demand assignment the contents of ODIJ may be adjusted, according to the variable demand algorithm, during each iteration and GPQRAN is updated to save the current values.

KD is used in the road traffic assignment to omit processing for origins for which the total demand is zero.

Initialisation

VCUM	is initially set in 6.60 RASS to the link volume for an all or nothing assignment
TRAV	is initially set in 6.60 RASS to the free flow travel time (time at zero link volume)
ODIJ	is refilled, as each origin is processed, with the current demand for that origin read from file GPQRAN
KD	is set in 6.60 RASS as follows: KD(IP) = IP if total demand from origin IP is non-zero KD(IP) = -IP if total demand from origin IP is zero.

References

6.60	RASS
6.61	LOADTR
6.62	MINTRE
6.63	OBJFN
6.64	OPSTEP
6.68	MOES
6.80	RASSAV
6.94	PRTLNK
6.95	PRTGPQ

COMMON / RVAR1 /

```
      COMMON /RVAR1/ FLOW(%LINK%),DIR(%CENT%),UPQ(%CENT%)
```

Description of variables

/RVAR1/ is used as a work area during a road traffic assignment.

Variable	Type	Description
FLOW	REAL	FLOW(L) contains the automobile volume for the all or nothing assignment made as part of the current iteration, L=1,NLINK.
DIR	REAL	DIR(IQ) contains the demand adjustment from the current origin to destination IQ for the current iteration, IQ=1,NCENT.
UPQ	REAL	UPQ(IQ) contains the automobile impedance from the current origin to destination (IQ), calculated as the inverse of the demand function, IQ=1,NCENT.

Use

All the variables in /RVAR1/ are used to store intermediate values calculated during the road traffic assignment. DIR and UPQ are used during variable demand assignments only.

Initialisation

Not applicable.

References

6.60	RASS
6.61	LOADTR
6.64	OPSTEP

COMMON / RVAR2 /

```
COMMON /RVAR2/ KPQAU(%CENT%),GTOT(%CENT%), KP(4)
REAL KPQAU,KP
```

Description of variables

Variable	Type	Description
KPQAU	REAL	KPQAU(IQ) contains the component of the demand function which is independent of automobile travel time, between the current origin and destination IQ (K_{pq}^{au})
GTOT	REAL	GTOT(IQ) contains the vehicle (auto) equivalent of the total demand, by both modes, from the current origin to destination IQ
KP	REAL	Demand function coefficient for the current origin KP(1) = k_0 KP(2) = k_1 KP(3) = k_2 KP(4) unused

Use

KPQAU	calculated in module 6.25 CALCK for use in road traffic assignment with variable demand
GTOT	used to contain the total demand from the current origin read from the file GBPQ, used in demand function calculations in a variable demand road traffic assignment
KP	used in any demand function calculations for variable demand road traffic assignment. Read from databank (file 2.31 DCOK).

Initialisation

Not applicable.

References

6.25	CALCK
6.60	RASS
6.61	LOADTR
6.63	OBJFN
6.64	OPSTEP
6.73	DEMCST
6.74	TDCST

COMMON / STRUC /

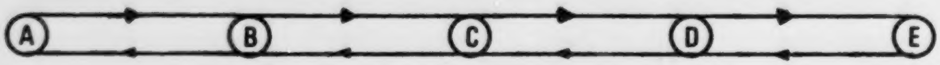
```
      COMMON /STRUC/ FLIN(%LINE%),DA(%SEG%),DR(%SEG%),
     +               DISA(%SEG%),DISR(%SEG%),
     +               NOLIST(%LINE%),NONLIN(%LINE%),
     +               LISTN(%SEG%),NDIR(%SEG%),
     +               LISTL(%SEL%),LPNT(%SEL%),NADR(%SEL%)
      INTEGER%*2% NOLIST,NONLIN,LISTN,NDIR,LISTL,LPNT,NADR
```

Description of variables:

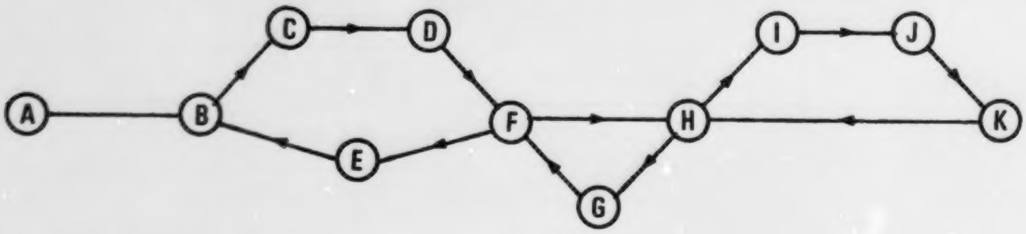
Refer to specification of modules 5.12 and 5.15 and files 501 and 515.

Use:

Common STRUC is a list structure representing the transit network An example with two transit lines will show how the lists are built.



line 1



line 2

LINE POINTERS

address	1	2
NOLIST	1	6
NONLIN	5	16

address	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LISTN	A	B	C	D	E	A	B	C	D	E	F	G	H	I	J	K
NDIR	0	0	0	0	0	0	0	+	+	−	0	−	0	+	+	0
DA	•••	AB	BC	CD	DE	•••	AB	BC	CD	•••	DF	•••	FH	HI	IJ	JK
DR	•••	BA	CB	DC	ED	•••	BA	•••	•••	EB	FE	GF	HG	•••	•••	KH

*node	*A	*B	*C	*D	*E	*F	*G	*H	*I	*J	*K					
address	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LISTL	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2
LPNT	12	13	14	15	16	0	0	0	0	0	0	0	0	0	0	0
NADR	1	2	3	4	5	11	12	13	14	15	16	6	7	8	9	10

Initialisation:

FLIN,NOLIST,NONLIN,LISTN,NDIR,LISTL,LPNT,NADR are defined by module 5.12 BLDNET.

DA,DR,DISA,DISR are defined by module 5.15 TTIME.

DA,DR are modified by module 4.31 TIMCHK.

Common STRUC is saved on file 515 TNET by calling subroutine WTNET. It is retrieved by calling subroutine RTNET.

References:

	TRANSIT (dummy module for CDC version only)
4.21	TIMPED
4.31	TIMCHK
4.61	CAPCHK
5.00	RTNET
5.01	WTNET
5.03	TDISCO
5.09	LINOUT
5.11	LISTLI
5.12	BLDNET
5.13	TIMOUT
5.14	EXPNET
5.15	TTIME
5.16	EQUIVA
5.17	EQUIVR
5.20	TPATH
5.21	ACCESS
5.22	FORLAB
5.23	REVLAB
5.31	ACCESX
5.32	FORLAX
5.33	REVLAX
5.50	TASS
5.56	LOASEC
5.60	FLOOUT

COMMON /SUPCOM/

```
COMMON /SUPCOM/ IMATOD,DBMOD,DCOSSW,ICERR,DEBUG,I15(15)
LOGICAL DBMOD,DCOSSW,DEBUG
```

Description of variables

Variable	Type	Description
IMATOD	INTEGER	Current OD matrix number (from parameter ODMA)
DBMOD	LOGICAL	TRUE indicates data bank has been modified in this execution of EMME
DCOSSW	LOGICAL	TRUE indicates module 7.10 DCOS01 has been called in this execution of EMME
ICERR	INTEGER	Number of EMME command errors
DEBUG	LOGICAL	Indicates last setting by $DEBUG. TRUE indicates $DEBUG,YES
I15	Undefined	Reserved space for future use.

Use

IMATOD	is used by the data bank routine PSOD to determine which OD matrix to read
DBMOD	is used to prevent more than one call to 1.02 MODBAN
DCOSSW	is used to prevent calling 7.10 DCOS01 a second time for a second assignment if data bank has not been modified
ICERR	counts command errors so that the system can be terminated if more than 10 errors are found
DEBUG	to switch debugging output on (TRUE) and off (FALSE).

Initialisation

IMATOD = 1 Set by $ASSIGN initialisation

DBMOD = FALSE Set by 1.10 INISUP

DCOSSW = FALSE Set by 1.10 INISUP

ICERR = 0 Set by 1.10 INISUP

DEBUG = FALSE Set by 1.10 INISUP

References

1.00	EMSUP
1.10	INISUP
1.15	DECDEB
1.30	DECCOM
1.60	DECASS
1.62	DECPAR
3.13	PSOD
3.27	PSOCC
3.28	PSMEST
3.29	PSODEX

COMMON / TRPATH /

```
COMMON /TRPATH/ TIAC(%NODET%),NOAC(%NODET%),LIAC(%NODET%)
```

Description of variables:

See description of file 5.21 TTREES.

Use:

Gives the tree of transit paths from one origin centroid to all destination nodes. Passengers are assigned on those paths.

Initialisation:

The tree is initialised in modules 4.21 TIMPED and 5.50 TASS. It is then calculated by module 5.20 TPATH.

Module 4.21 TIMPED saves the trees on file TTREES and the trees from the last iteration are then read directly from the file in the case of a bimodal assignment. This prevents performing the same path calculations twice (see module 5.50 TASS).

References:

4.21	TIMPED
5.20	TPATH
5.21	ACCESS
5.24	LABEL
5.31	ACCESX
5.32	FORLAX
5.33	REVLAX
5.50	TASS
5.56	LOASEC

COMMON / TRWALK /

```
 COMMON /TRWALK/ NAC(%CENT+1%),NEG(%CENT+1%),NODAC(%TACCES%),
+                ACT(%TACCES%),NODEG(%TACCES%),EGT(%TACCES%),
+                ACFLO(%TACCES%),EGFLO(%TACCES%)
```

Description of variables:

Variable	Type	Description
NAC (I)	INTEGER	address in NODAC of transit access nodes for centroid I
NEG (I)	INTEGER	address in NODEG of transit egress nodes for centroid I
NODAC	INTEGER	list of transit access nodes (destination node of access links)
ACT	REAL	user-defined walk times on transit access links
NODEG	INTEGER	list of transit egress (origin node of egress links)
EGT	REAL	user-defined walk times on transit egress links
ACFLO	REAL	predicted volumes on transit access links
EGFLO	REAL	predicted volumes on transit egress links

Use:

To calculate transit shortest paths and save predicted access-egress volumes in the transit assignment. The labelling process of transit path calculations is performed in three stages:

1) label all access nodes, using access links
2) continue the labelling process by considering only the network nodes
3) complete the paths by selecting the egress links which yield the smallest trip times

Initialisation:

Module 5.50 TASS reads access and egress links from files ACCESS and EGRESS and calculates access and egress volumes which are saved in ACFLO and EGFLO.

References:

5.50	TASS
5.60	FLOOUT

COMMON /UNIT/

```
 COMMON /UNIT/ BANK  ,CORNET,DCOKAU,DCOK01,DCOKTR,DIRFIL,GBPQ   ,
+              GPQAU ,GPQRAN,GPQTR ,RLTIVL,RZTIM ,TFLOW ,TLFREQ,
+              TZIMP ,TZTIM ,TNET,  CODFIL,COMFIL,PCHFIL,TTREES
 INTEGER BANK  ,CORNET,DCOKAU,DCOK01,DCOKTR,DIRFIL,GBPQ   ,
+        GPQAU ,GPQRAN,GPQTR ,RLTIVL,RZTIM ,TFLOW ,TLFREQ,
+        TZIMP ,TZTIM ,TNET,  CODFIL,COMFIL,PCHFIL,TTREES
```

Description of variables

Each variable in /UNIT/ specifies the logical unit number of the file which has the same name as the variable. With the exception of BANK, the file number can be found in Appendix D, the corresponding file description in Appendix C.

The logical unit BANK consists of all the files numbered 200, 201 etc. and 300, 301, etc. (see section 5. for details of data bank structure).

Use

Each variable in /UNIT/ is used in any input/output statement referring to the file with the same name.

Initialisation

All the variables in /UNIT/ are set by the system BLOCK DATA subprogram.

References

1.01	BLOCK DATA
1.02	MODBAN
1.11	RDCOM
3.63	UCEN
3.64	UGEO
3.65	ULINE
3.66	UMAT
3.67	UMINI
3.68	URL
3.69	USCEN
3.70	UTL1
3.72	UZONEG
3.73	UDFTTC
3.74	UTPM
3.76	USE
4.15	RDLTV2
4.18	BUSFRQ
4.21	TIMPED
4.31	TIMCHK
4.61	CAPCHK
5.00	RTNET
5.10	WTNET
5.13	TIMOUT
5.14	EXPNET
5.15	TTIME

5.30	CALKTR
5.41	ODTRAN
5.50	TASS
5.55	TDEM
5.60	FLOOUT
5.90	CLAPRT
5.91	PRTLFR
5.92	PRCOR
5.94	PRIMPE
8.01	PTITLE
8.09	BLOCK DATA
8.43	TABLE
8.62	DINTRN
8.82	DINTRV
8.83	VOLACC

Appendix B: Module specifications

This appendix contains the specifications of the most important modules in the EMME system. The functions of certain modules are described within the specifications of other modules. The location of the description (if any) for a given module can be determined from the cross reference list in Appendix D.

The module specifications are arranged in module number order.

Module : 0.01 GMEM

Purpose : allocation of free memory

Called from : almost every module

input/output : 3 parameters

N
- = 1 reserve an area
- = 2 return an area
- = 3 reserve more memory for a given area
- = 4 return part of an area

NB: number of words to be reserved or returned

loc: position in common /MEM/ where area begins (may be input or output parameter)

matrix used: NOTE (100,3)

the module can keep track of 100 areas

NOTE (i,1) contains address of loc

NOTE (i,2) contains number of words in the area

NOTE (i,3) contains position of the area in common /MEM/

procedure:

a) case N = 1
- . if there is not sufficient space at end of memory, execute step e) as a closed subroutine
- . insert the entry in matrix NOTE.

b) case N = 2
- . find the address of loc in matrix NOTE
- . if this is the highest area in memory, update the variable NEXT; in other cases, eliminate the entry in NOTE.

c) case N = 3
- . find the address of loc in matrix NOTE
- . if there is not sufficient space at end of memory, execute step e) as a closed subroutine
- . move upward any area listed above this one
- . keep track of any change in matrix NOTE
- . using the address of the pointers to the areas indicate everywhere in core that a move occured.

d) case N = 4
- . find the address of loc in matrix NOTE
- . the number of words to cancel should not be greater than the length of the area
- . if this is the highest area in memory, update the variable NEXT; in other cases shorten the length of the area in matrix NOTE.

e) garbage collection
- start with the lower area
- repeat the following as many times as there are removed areas
- find the area located just over it
- move it downward
- update the pointer to the area somewhere in core
- consider the moved area as the lower. At the end, check if the collected words are sufficient; if not stop the system.

MODULE : 1.00 EMSUP

PURPOSE : Initialize EMME system and decode commands

INPUT : File name : COMFIL

Type : Direct Entry Input File

Status : Always required

Contents : EMME commands, subcommands and data

Sequence : Sequential

Source : Supplied by user

PROCEDURE : In the following description, the names in the left hand column are subroutines which contain code for the section described to the right.

1) INISUP	Initialize system.
2) RDCOM	Read first command card. If EOF, print message and terminate EMME.
3) DECCOM	Print command if not already printed. Decode command
4)	If command is erroneous, add 1 to error count. Terminate EMME after 10 bad commands.
5)	Call subroutines to continue processing as follows:

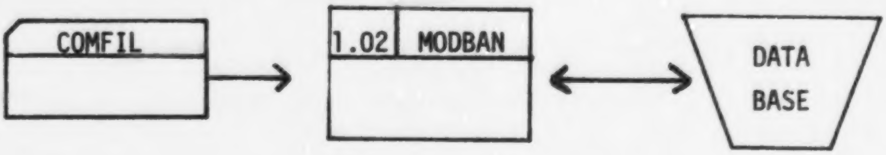
#	Command	Action
1	$END	Terminate EMME.
2	$COMMENT	Go to step 2.
3	$DATABANK	If this is second $DATABANK, terminate EMME. Otherwise CALL MODBAN to process headers and data.
4	$ASSIGN	CALL DECASS to decode $ASSIGN options and subcommands (See 1.60 DECASS). CALL OPBAN to open data bank if it is not already open (See 3.21 OPBAN).

#	Command	Action
		CALL DCOSO1 to compute socio-economic part of demand function (See 7.10 DCOSO1) for first variable demand or bimodal assignment. CALL ASSPRO to perform assignment (See 4.00 ASSPRO).
5	$REPORT	CALL OPBAN to open data bank if it is not already open (See 3.21 OPBAN). CALL DECREP to decode $REPORT subcommands.
6	$TITLE	CALL DECTIT to process title line text (See 1.40 DECTIT).
7	$DEBUG	CALL DECDEB to decode $DEBUG options (See 1.15 DECDEB).

6) go to step 2.

MODULE : 1.02 MODBAN

PURPOSE : Process $DATABANK subcommands, headers and data



INPUT : a) File name: COMFIL

Type	:	Direct Entry Input File
Status	:	Always required
Contents	:	EMME commands, subcommands and data
Sequence	:	Sequential
Source	:	Supplied by user

b) Data in COMMON area /CARTE/

INTEGER M(80)	$DATABANK command

c) Data bank files as described in the modules called by MODBAN

OUTPUT : a) Subroutine parameter

LOGICAL OK	TRUE if no catastrophic errors are found, FALSE otherwise.

b) Data bank file as described in the modules called by MODBAN

c) File name: OUTPUT

Type	:	Report file
Status	:	will be routed to printer
Contents	:	Listing of input from COMFIL and error messages
Sequence	:	Sequential

PROCEDURE : In the following description names in the left hand column are subroutines which contain code for the section described to the right. Not all the subroutines called by MODBAN are identified in the left hand column.

Process $DATABANK command and subcommand:

1.	ANCOM	Recheck for $DATABANK. If not, print error message and return.
2.	ANCOM	Check for option NEW.
3.		If NEW initialize data bank index.
4.		If not NEW, read existing index from data bank.
5.		Read next subcommand from COMFIL. If EOF or COMFIL go to step 32.
6.		Decode $DATABANK subcommand. If not fixed, go to step 32.

Process CORRECTION headers and data:

7.		If not CORRECTION go to step 14.
8.	ANCOM	Check LIST/NOLIST option.
9.		Read header
	HEADER	Decode header If not a header go to step 6.
10.		If OD MATRIX, pick up and check name if any.
11.	ANCOM	Check LIST/NOLIST option on header.
	B	Clear up the section of data bank if necessary.
12.	[Table]	Call routine to read, validate and process data following header according to table in Figure 1.02-1: CORRECTION subroutine.
13.		Complete processing of data following header. Go to step 9.

Process PURGE subcommand

14.		If not PURGE go to step 16.
15.		Process PURGE subcommand. Go to step 5.

Process SCENARIO subcommand

16.		If not SCENARIO go to step 20.
	SKNSEP ANCOM	Pick up name, decode LIST/NOLIST.
17.	SCENO	Register name, set accumulate scenario data.
18.		Read header
	HEADER	Decode header. If OD matrix, pick up and check name if any. Check LIST/NOLIST on header
	ACCUM	Accumulate header.
19.	[Table]	Call routine to read, validate and accumulate records following header according to table in Figure 1.02-1: CORRECTION subroutine.
	SCENF	Accumulate scenario section END card Go to step 5.

Process USE subcommand

20.		If not USE go to step 27.
	SKNSEP	Pick up name of scenario to use
	ANCOM	Check LIST/NOLIST option.
21.	SCEN1	Find scenario data and put in scenario stack. If not defined, set OK=FALSE, print message and go to step 5.
22.	SCEN2	Initialize access to scenario at top of stack.
23.	MUCCA	Get a scenario header.
	HEADER	Decode header. If 'INCLUDE' go to step 21 If end of scenario go to step 22.
24.	B	Clear up sections of data bank as required for header.
25.	[Table]	Call routine to process data following scenario header according to table in Figure 1.02-1: SCENARIO subroutine. Complete processing of data following header. (Same code as for CORRECTION data with same header, see step 13.)

```
26.             Go to step 23.

Process PRINT subcommand

27.             If not PRINT go to step 29.
    PRINT       Print data as specified in subcommand.

28.             Go to step 5.

Process PUNCH subcommand

29.             If not PUNCH, print subcommand error message and go
                to step 32.

30. PUNCH       Punch data as specified in subcommand.

31.             Go to step 5.

Terminate $DATABANK processing (Rewrite index + renumber)

32.             If databank not modified, return.

33. IJ          Sort, renumber and validate sections of the data bank,
                depending on modifications made (vector BETA).
                Set NOGO switch if catastrophic errors found.

34. KEEPND      Save node tables.

35. RWIND       Rewrite index

36.             Return.
```

NOTE: The comments in the module code contain 'command' instead of 'subcommand' and 'subcommand' instead of 'header'.

Headers	CORRECTION subroutine	SCENARIO subroutine
CENTROIDS	RCEN	OCEN
NODE EXPANSION	RMINI	OMINI
ROAD LINKS	RRL	ORL
TRANSIT LINKS	RTL	OTL
LINES	RLINE	OLINE
O-D MATRIX n	RMAT	OMAT
VEHICLE ADJUSTMENT MATRIX	RMAT	OMAT
AUTO OCCUPANCY	RMAT	OMAT
MODAL SPLIT	RMAT	OMAT
ZONE GROUPS	RZONEG	OZONEG
SOCIONOMIC VARIABLES FORMAT 1	RSE1	OSE1
SOCIONOMIC VARIABLES FORMAT 2	RSE2	OSE2
SOCIONOMIC COEFFICIENTS FORMAT 1	RSE1	OSE1
SOCIONOMIC COEFFICIENTS FORMAT 2	RSE2	OSE2
DEMAND FUNCTION COEFFICIENTS	RDFTTC	ODFTTC
NODE COORDINATES	RGEO	OGEO
SCALE	SCALE	SCALE
TRANSIT PATH MULTIPLIERS	RTPM	OTPM

Figure 1.02-1 Subroutine table

MODULE : 1.12 ANWORD

PURPOSE : Search table for given word

INPUT : a) Data contained in COMMON area /CARTE/

COL(80) contains the word, 1 character per element

b) Subroutine parameters:

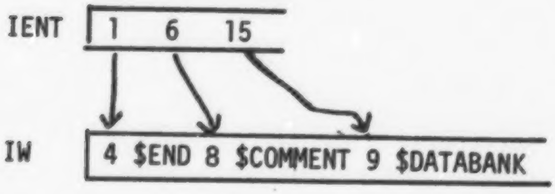
INTEGER ISTART	Position in COL(80) to start scan for word.
INTEGER IDENT	Vector containing list of entries in IW which are valid for this search operation.
INTEGER LENT	Number of entries.
INTEGER ISEP1	- Comma (1H,) if separators before word can be blank or comma. - a character (1Hx) if separator before word must be character x.
INTEGER ISEP2	Same as ISEP1 but applies to separator after word.
INTEGER IW	Vector containing a variable length list of words allowed. Description of structure follows.

IW contains the list of words. Each word is specified as its length followed by the word itself, 1 character to a vector element. For example words $END, $COMMENT, $DATABANK would be entered as

```
DATA IW / 4, 1H$, 1HE, 1HN, 1HD
          8, 1H$, 1HC, 1HO, 1HM, 1HM, 1HE, 1HN, 1HT,
          9, 1H$, 1HD, 1HA, 1HT, 1HA, 1HB, 1HA, 1HN, 1HK /
```

The corresponding entries in IENT would be

```
DATA IENT/1, 6, 15/
```



OUTPUT : Subroutine parameters:

INTEGER IFIRST	Position in COL(80) of the first non separator found in the scan starting from position ISTART. If only separators are found, IFIRST is set to zero.
INTEGER IEND	Position in COL(80) of the first separator found in the scan after position IFIRST. If no word was found (IFIRST is zero), IEND is not changed by ANWORD.
INTEGER NCOM	If for the word found in COL(80) there is an identical word found in IW, NCOM contains the entry number in IENT for that word. NCOM is zero if the word is not found.
LOGICAL LEOR	FALSE if a word was found in COL(80). TRUE if the position 72 of COL(80) was reached and a word was not found i.e. COL(80) contains separators only from position ISTART on. (Logical end of record).

PROCEDURE : In the following description, the names in the left hand column are subroutines which contain code for the section described to the right.

1) SKACH	If ISEP1=1Hx and x≠comma, skip any character x preceding word (separator is x).
SKSEPC	If ISEP1=comma, skip blanks or commas preceding word (separator is blank or comma).
2)	If only separators were found in step 1, set LEOR=TRUE and return.
3)	Count the characters in the word in COL(80):
SKNACH	If ISEP2=1Hx, scan until character x is found. If ISEP2=comma, scan until a blank or comma is found.

4) For each entry number in IENT, until search word in COL(80) is found in IW:

- pick up length from IW.
- set comparison length to minimum of characters in search word and length from IW but at least 3.
- compare search word and word in IW.

5) set NCOM equal to entry number in IENT if word was found, zero otherwise.

6) Return.

MODULE : 1.15 DECDEB

PURPOSE : Decode $DEBUG command option

INPUT : a) Data in COMMON area /CARTE/

COL(80) contains the command $DEBUG, 1 character per element

b) Subroutine parameter

INTEGER ICOL	Position of first separator following the command code $DEBUG

OUTPUT : a) Data in common area /SUPCOM/

LOGICAL DEBUG	TRUE if option is YES FALSE if option is NO

b) Subroutine parameter

LOGICAL OK	TRUE if option can be decoded FALSE if option is invalid

PROCEDURE : In the following description, the names in the left hand column are subroutines which contain code for the section described to the right.

1) SKSEPC	Skip separating blanks or commas in COL(80) starting from ICOL.
2) GTWORD	Pick up option following separators (See 1.63 GTWORD).
3)	Check option for YES or NO and set DEBUG as described in OUTPUT. Set OK=TRUE. If option is not YES or NO, print a message and set OK=FALSE.
4)	Return.

MODULE : 1.30 DECCOM

PURPOSE : Decode command code

INPUT : a) Data in COMMON area /CARTE/

INTEGER COL(80)	contains the command, 1 character per element.
LOGICAL PRINTD	Print indicator - TRUE if command has been printed - FALSE if command has not been printed.

b) Subroutine parameters

INTEGER ISTART	Position in COL(80) where scan for command is to start.

OUTPUT : Data in COMMON area /CARTE/

LOGICAL PRINTD	Set to TRUE when command printed.
INTEGER IEND	Position of first separator following command code.
INTEGER NCOM	Number of command code decoded.

PROCEDURE : In the following description the name in the left hand column is the subroutine containing the code for the section described to the right.

1.	Print command if not already printed.
2. ANWORD	Pick up command code and search for it in command table IW. For format of IW see 1.12 ANWORD
3.	If command is unknown, print message and set OK to FALSE.
4.	Return.

MODULE : 1.40 DECTIT

PURPOSE : Save title time from command $TITLE

INPUT : a) Data in COMMON area /CARTE/

INTEGER COL(80) $TITLE command

b) Subroutine parameter

INTEGER ICOL	Position in COL(80) to start scan for title line number.

OUTPUT : a) Data in COMMON area /REP/

INTEGER TITLE (60,2)	Title line text, 60 characters for 2 lines, 1 character per element.

b) Subroutine parameters

LOGICAL OK	FALSE if title line number not a valid number. TRUE otherwise.

PROCEDURE : In the following description, the names in the left hand column are subroutines which contain code for the section described to the right.

1. GTWORD	Pick up title line number.
2.	If no line number is found, print a message and return. Title line text is unchanged.
3.	If line number is not 1 or 2, set to 1.
4.	If first character of title text is blank, move 60 characters maximum to title line.
5.	If first character of title text is non blank, count characters (60 max or to end of record) and centre text in title line.
6.	Print new contents of title line.
7.	Return.

MODULE : 1.60 DECASS

PURPOSE : Decode and set assignment options. Decode subcommands

INPUT : a) Data contained in COMMON area /CARTE/

INTEGER COL(80) $ASSIGN command

b) Input file

File name : COMFIL

Type : Direct entry input file

Status : Always required

Contents : EMME commands and subcommands

Source : Supplied by user.

OUTPUT : Data contained in COMMON area /APARM/

Variables in /APARM/ are set according to the $ASSIGN options as shown in Table 1. Decoding of parameters in PARAMETERS subcommand is described in module 1.62 DECPAR.

PROCEDURE : In the following description, the names in the left hand column are subroutines which contain code for the section described to the right.

1.	Initialize variables in APARM to default values (see Appendix 1.62 DECPAR).
2. ANWORD	Decode next option on $ASSIGN command using specified part of option table IENTCP.
3.	If end of command record go to step 5.
4.	Adjust part of option table to valid options. Go to step 2.
5.	(All options decoded). Initialize ASSPSW in /APARM/ according to type of assignment as shown in table 1. Set default values of other variables when default depends on type of assignment.

6.		Read subcommand into COL(80).
7.		Skip leading blanks If command ie no more subcommands, go to step 14.
8.	ANWORD	Decode subcommand code. NCOM = code number.
9.		If subcommand unknown, print message and go to step 6.
10.		Go to routine to process subcommand IDENT step 11 PARAMETERS step 12
11.		Process IDENT subcommand. - Skip blanks preceding identification text. - Pack 40 characters identification text into IDEN in /APARM/. Go to step 6.
12.	DECPAR	Decode keyword = value pairs and place values in /APARM/. See 1.62 DECPAR.
13.		Go to step 6.
14.		(Command or end of file on command file). Check compatibility of /APARM/ variables.
15.		Return.

Table 1. ASSPSW switch setting: for different assignment types.

Switches		Options				
#	Name	BIMODAL	TRANSIT FIXED	TRANSIT VARIABLE	ROAD FIXED	ROAD VARIABLE
1	TNETSW	T	T	T		T
2	BUSFSW	T				
3	TIMPSW	T				T
4	ROADSW	T			T	T
5	VELTSW	T				
6	TRANSW	T	T	T		
7	VECPSW					
8	TVARSW			T		
9	BIMOSW	T				
10	RVARSW	T				T
11	BUSISW	T				

MODULE : 1.62 DECPAR

PURPOSE : Decode parameter of PARAMETERS subcommand

INPUT : a) Data in COMMON area /CARTE/

INTEGER COL(80) contains the PARAMETERS subcommand, 1 character per element.

b) Data in COMMON area /APARM/

LOGICAL BIMOSW	TRUE for bimodal assignment, FALSE otherwise.
LOGICAL TVARSW	TRUE for transit-only variable demand assignment, FALSE otherwise.
LOGICAL RVARSW	TRUE for road-only variable demand or bimodal assignment, FALSE otherwise.

c) subroutine parameters.

INTEGER ICOL	Position in COL(80) where scan for first keyword = value pair is to start.

OUTPUT : a) Data in COMMON area /APARM/

Variables in /APARM are set according to the keyword = value pairs decoded. See Appendix "EMME assignment parameters handling". (1.62-3)

b) Data in COMMON area /SUPCOM/

INTEGER IMATOD	Set by parameter ODMA.

c) Subroutine parameters

LOGICAL OK	Set to FALSE if any parameter errors are detected. Unchanged otherwise.

PROCEDURE : In the following description, the names in the left hand column are subroutines which contain code for the section described to the right.

1. ANWORD Find and decode next keyword. NPAR set to keyword number (see 1.12 ANWORD).

2. If end of record found in step 1, return.

3. If keyword unknown i.e. not in table IWP (NPAR=0), print message, skip following value. Go to step 1.

4. GTWORD Get value following equal sign (=). (See 1.63 GTWORD).

5. Validate value according to keyword (table in IV).

    1 = must be numeric
    2 = must be YES or NO
    3 = no validation

   Print message if invalid and go to step 7.

6. Place value according to keyword, validating previously unvalidated values.

7. Advance to next keyword. Go to step 1.

NOTES: The format of IWKEY, the list of keywords allowed, and IWP the list of entries in IWKEY are described as subroutine parameter IW and IENT in module 1.12 ANWORD.

Appendix : EMME assignment parameter handling

Keyword = Value	Variables to be set		Default	
1. ITLT=xxxx	MXLT=xxxx		Default: except bimodal	bimodal
	VELTSW = FALSE if xxxx=0		MXLT = 1	=1
	= TRUE if xxxx≠0		VELTSW = F	=T
2. ITCP=xxxx	MXCP=xxxx		Default: MXCP = 0	
	VECPSW=FALSE if xxxx=0		VECPSW = F	
	VECPSW=TRUE if xxxx≠0			
3. ITRD=xxxx	MXRD=xxxx		Default: 15	
4. ITCK=xxxx	Not implemented, keyword invalid.			
5. RTIM=xxxx	TMAX=xxxx		Default: 0.0	
6.	USEPLT			
LTIM=OBS	FALSE		Default: FALSE	
=PRED	TRUE			
7.	SPLT	SPLTV		
SPLT=.xxx	VALU	.xxx		
=PCNT	PCNT	——		
=PERS	PERS	——		
=PRED	PRED	——		
=AUTO	AUTO	——		
	DEFA	——	Default, value used = %SPLTD%	

Keyword = Value	Variables to be set				
8.	AOCC	AOCCV			
AOCC=x.xx	VALU	x.xx			
=PERS	PERS	——			
	DEFA	——			Default, value used = %AOCCD%
9.	BUSFSW	BUSISW	BUSFP	TNETSW	
BUSF=NO	FALSE	FALSE	NO	no change	<- Default(road var,fix)
=OBSN	FALSE	TRUE	OBSx	no change	
=COMP	TRUE	TRUE	COMP	TRUE	<- Default (bimodal)
10. CONT=YES	Not implemented, keyword invalid.				
=NO					
11. ODAU=YES	Not implemented, keyword invalid.				
=NO					
12. DSPD=xx.x	TDSPED=xx.x				Default: 10.0
13.	DFNC				
DFNC=LOGI	1				Default: 1
=LINE	2				
14. DTIM=xx.x	TDTIME=xx.x				Default: 15
15. TOLT=.xxx	EPSTIM=.xxx				Default: .05
16. TOLU=.xxx	TOLERZ=.xxx				Default: .05
17. WFAC=.xxx	WFACT=.xxx				Default: .5
18. WPEN=xx.x	WPENAL=xx.x				Default: 0.0

Keyword = Value	Variables to be set		
19. WMIN=xxx.	WAITMI=xxx.		Default 0.0
20. WMAX=xxx.	WATIMA=xxx.		Default: 120.0
21.	VALID		
TVAL=YES	TRUE		Default: TRUE
=NO	FALSE		
22. BCAP=xxx.	TVCAP=xxx.		Default: 60.0
23. HDWY=xxx.,...	HDWSET(1)=xxx. etc.		Default: 2.,5.,10.,15.,20.,30.
	MAXHDW=Number values found.		
24. ODMA=x	IMATOD=x		Default: 1
=name	IMATOD=matrix number corresponding to name.		
25. BEQU=x.xx	BUSFAC=x.xx		Default: 3.0
26. EXPF=x.xx	ALFEXP=x.xx		Default: .75
27. VADL=xxxx	VADLIM=xxxx		Default: 50
28.	TNETSW	TIMPSW	
TIMP=COMP	TRUE	TRUE	<- Default(road, var only).
=PRED	no change	FALSE	
29.	LINDIV		
TDIV=YES	TRUE		Default: FALSE
NO	FALSE		
30.	USEWP		
PMUL=YES	TRUE		DEFAULT: False
NO	FALSE		

Keyword = Value	Variables to be set	
31. WALK=x.xx	WWALK=x.xx	Default: 1.0
32. WAIT=x.xx	WWAIT=x.xx	Default: 1.0
33.	ODEXSW	
VEAD=YES	TRUE	Default: False
NO	FALSE	
34. TOLV=.xxx	EPSIL=.xxx	Default: .05

MODULE : 1.63 GTWORD

PURPOSE : Gets next word from subcommand record.
Numeric values are decoded.

INPUT : a) Data contained in COMMON area /CARTE/

INTEGER COL(80) contains the word, 1 character per element.

b) Subroutine parameters:

INTEGER LOCN Position in COL(80) to start scan for word.

OUTPUT : Subroutine parameters:

LOGICAL OK	FALSE if numeric value contains more then 7 digits. TRUE otherwise.
INTEGER IEND	Position in COL(80) of the first separator found in the scan after word. If no word was found (LEOR=TRUE) IEND is not changed by GTWORD.
INTEGER CHAR	Set to first 4 characters of word if one is found. Format A4. if less than 4 characters are found, blanks are added on the right.
REAL VALUE	If word specifies a valid numeric value (see below), VALUE is set to the value, otherwise VALUE is not changed by GTWORD.
LOGICAL NIND	If VALUE is set to a value, NIND is set to TRUE, otherwise it is set to FALSE.
LOGICAL LEOR	(Logical end of record) TRUE if position 72 of COL(80) is reached and no word was found, i.e. COL(80) contains only blanks from LOCN on. FALSE if a word was found.

PROCEDURE : In the following description the names in the left hand column are subroutines which contain code for the section described to the right.

1. SKACH Skip any blanks preceding word.

2. If only blanks are found in step 1, set LEOR=TRUE and return.

3. SKNSPC Count the characters in word in COL(80). Scan until a blank or comma is found.

4. Ckeck first character for + or - sign. Save sign if any.

   Scan characters for valid sequence of digits and decimal point. Any other character stops scan and sets NIND to FALSE. Second decimal point stops scan and set NIND to FALSE. (See below for conversion algorithm).

   First 4 characters saved for packing.

5. If valid number sequence, convert to real number in VALUE and set NIND to TRUE.

6. Pack 4 characters to CHAR.

7. Return.

NOTES: Valid numeric value and conversion.

A valid numeric value may contain:

- maximum 7 digits (including explicit zeros)
- preceding plus or minus sign
- 1 decimal point.

The conversion is accomplished as follows:

- The string of sign, digits and possible embedded decimal point is scanned from left to right.
- Any sign is saved.
- Digits are accumulated from left to right using the formula
  Accumulator=(Accumulator x10) + new digit.
  for each digit encountered.

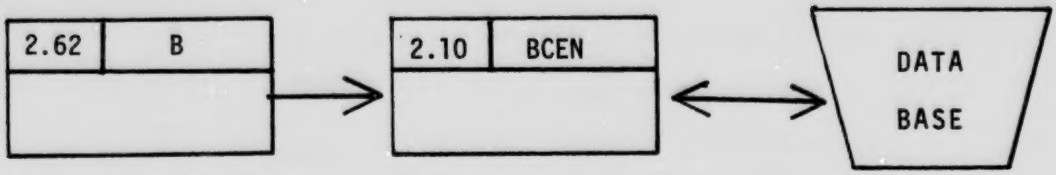
- Once a decimal point has been found, the number of digits following the point are counted and the digits are added to the accumulator as already described. When the last digit has been processed the real value of the number is calculated as

  Real value=Accumulator / $(10^{\text{number of digits after point}})$.

- The real value is made negative if a negative sign was found.

Module : 2.10 BCEN

Purpose : clear up file of centroids



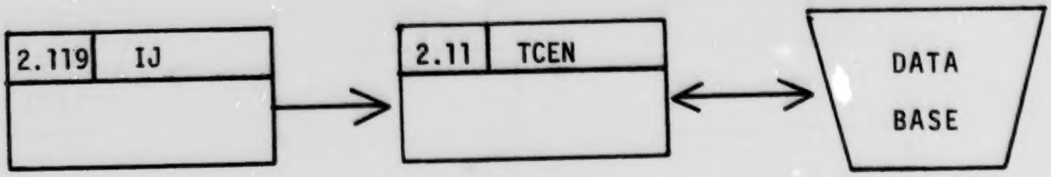
input: see module 2.11 TCEN

output: same file as input

procedure:
a) read each block of the file
b) erase each record added temporarily or purged permanently; reactivate each record temporarily deactivated
c) set new length of the block and rewrite it.

Module : 2.11 TCEN

Purpose: sort centroids according to external numbers



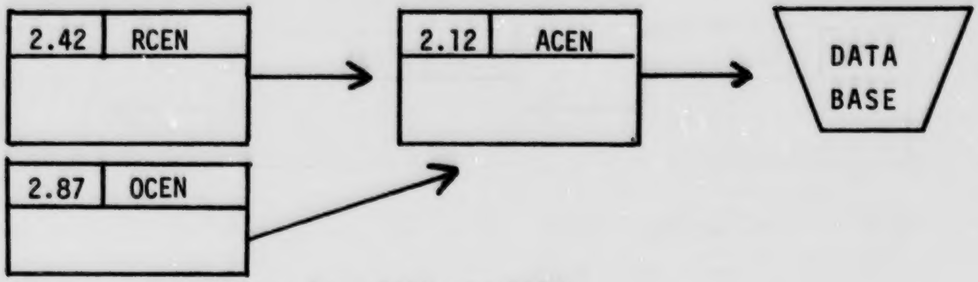
input: file name: CENT (file of centroids)
type: data bank
contents: data describing each centroid
sequence: sequential
created by: 2.12 ACEN
source: user specified

output:
1. same file as input
2. a vector containing external numbers of centroids, sorted

procedure:
a) reserve an area of 8 words per centroid
b) transfer to core all active centroids
c) reserve an area for the numbers of the centroids and another for their position in core
d) if centroids have already been sorted in the history of the file, use a bubblesort
e) if they have never been sorted, use a quicksort
f) put back sorted centroids on file
g) keep in core the vector of sorted numbers.

Module : 2.12 ACEN

Purpose : add a new centroid to data bank



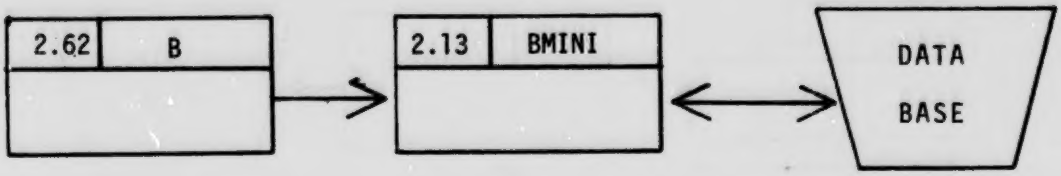
input : Parameters from RCEN and OCEN
NUMBER of the centroid
X - coordinate
Y - coordinate
TEXT containing name of zone (dimensioned to 16)
IACT $\begin{cases} = 0 & \text{if permanent correction} \\ = 1 & \text{if temporary modification} \end{cases}$

output : file name: CENT (file of centroids)
type: data base
status: always modified
contents: data describing each centroid
sequence: sequential
source: user specified

procedure : a) check if already defined
b) add the new centroid at the end of file

Module : 2.13 BMINI

Purpose : clean up the correspondence tables



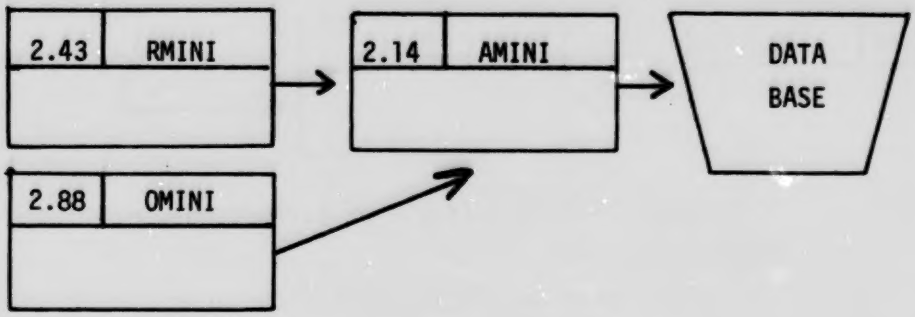
input: file name: NODCOR (table of correspondence)
type: data base
contents: data describing each expansion

output: same file as input

procedure:
a) read the file, record by record
b) transfer to another record all active expansions or expansions temporarily deactivated
c) erase the internal number of nodes
d) rewrite the clean records

Module : 2.14 AMINI

Purpose: add a new list to the correspondence table



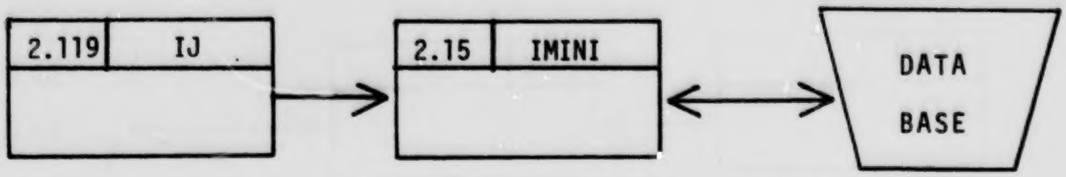
input : Parameters from RMINI and OMINI
NODE at center of expansion
NBEXPL, the number of nodes in the expansion
IEXPL, vector containing each node in the expansion
IACT $\begin{cases} = 0 & \text{if permanent correction} \\ = 1 & \text{if temporary modification} \end{cases}$

output : file name: NODCOR (table of correspondence)
type: data base
status: always modified
contents: data describing each expansion
sequence: sequential
source: user specified

procedure : a) check if already defined
b) add new record at end of file

Module : 2.15 IMINI

Purpose : assign internal numbers to the nodes in the correspondence table.



input:

1. file name: NODCOR (table of correspondence)
   type: data base
   contents: data describing each expansion
2. vector of sorted turning nodes
3. vector of road-only nodes
4. vector of sorted transit-only nodes

output: same file as input

procedure:

a) read the file, record by record

b) check if the node at center of expansion is a transit-only node. Multiply the external number by 2^{14} and add the internal number

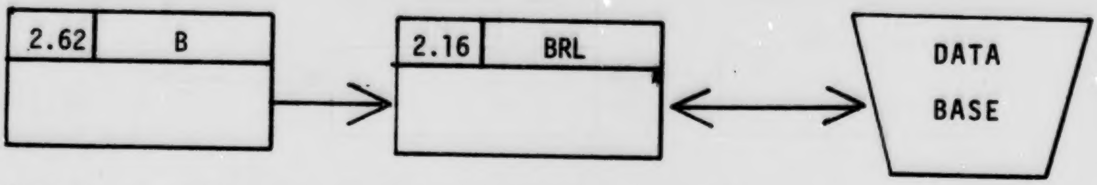
c) consider each node in the expansion;
   .if there is only one such node, it must be either 0 or a regular road node
   .if there are many nodes in the expansion, each of them must be a turning node

d) multiply the external number by 2^{14} and add the internal number

e) rewrite the record.

Module : 2.17 BRL

Purpose : clean up file of road links and prepare next modifications



input: file name: RLNKIN (file of road links)
type: data base
contents: definition of road links

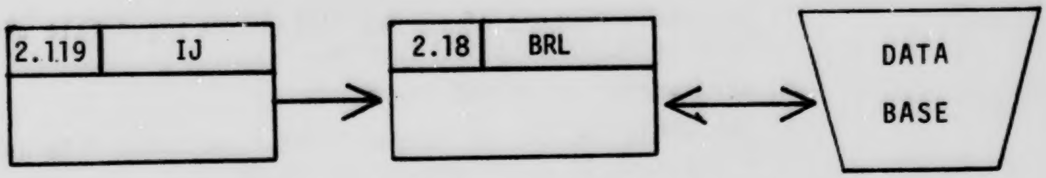
output: 1. same file as input

2. A list of known road links with their position in the file

procedure:
a) reserve an area for the list of links
b) read the file, by block
c) keep only active links and links deactivated temporarily
d) keep their origin, destination and position in file
e) set new length of the block and rewrite it.

Module : 2.18 SRL

Purpose : separate road links in 3 sections, assign internal numbers and sort.



input:
1. file name: RLNKIN (file of road links)
   type: data base
   contents: definition of road links
   source: user specified
2. vector of sorted centroid numbers

output:
1. file name: RLINK (file of links in network)
   type: data bank
   contents: description of road links
   sequence: sequential in 3 sections
   source: from the input file
2. vector of turning nodes, sorted
3. vector of road nodes.

procedure:
a) read the input file
b) check if each node is a turning node, using function TURLIN
c) construct the vector of turning nodes
d) sort the vector
e) read again the input file
f) determine if the link is an access, egress or normal link, using vector of centroids, vector of turning nodes and constructing at same time vector of road nodes.
g) transform the origin and destination to internal numbers and keep them, with position of the record in input file, in one of the 3 vectors being constructed (access, egress, normal)
h) erase file RLINK if one exists

i) sort vector of access links

j) read randomly the input file to get the whole record of an access link and write this record in second section of RLINK

k) sort vector of egress links

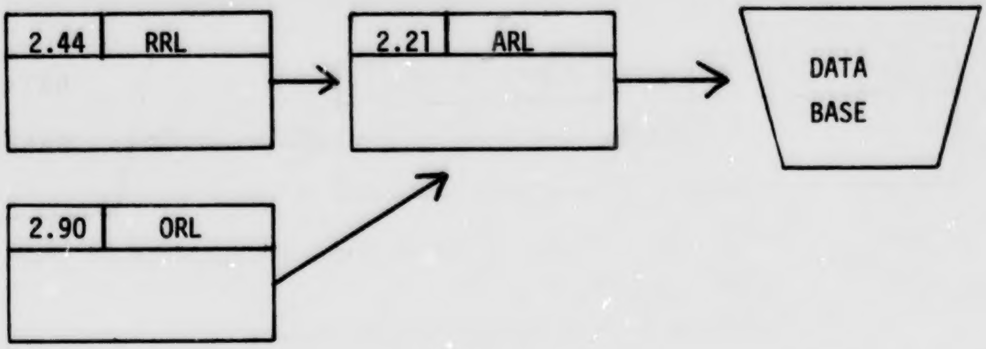
l) get whole records of egress links from input file and write them in RLINK as third section

m) sort vector of normal links

n) get whole records of normal links from input file and write then as first section of RLINK.

Module : 2.21 ARL

Purpose : add a road link to data bank



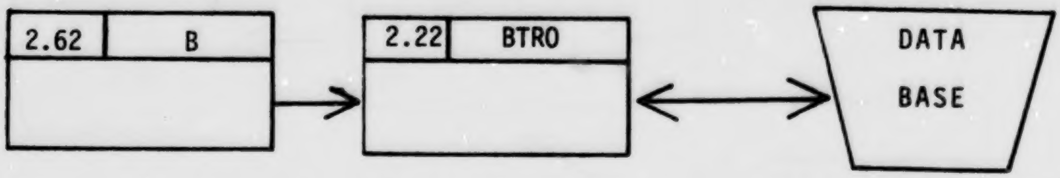
input:
1. parameters from RRL and ORL
   origin node
   destination node
   type of link
   length
   observed time
   observed volume
   5 user defined data
   text (16 characters)
   IACT = 0 if permanent correction
   IACT = 1 if temporary modification
2. the list of known road links and the position of the records on file, prepared by BRL

output:
file name: RLNKIN (file of road links)
type: data base
status: always modified
contents: definition of road links
sequence: random
source: user specified

procedure:
a) find the link in the list of known links. It must be inactive or nonexistent
b) add the new link to file
c) update or create a pointer in the list.

Module : 2.22 BTRO

Purpose : clean up file of nodes encountered on transit lines



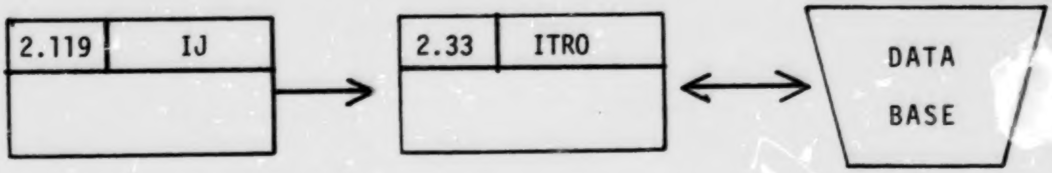
input: file name: STOPS (file of nodes encountered on transit lines)
type: data base
contents: node numbers and frequency

output: same file as input

procedure:
a) read each block
b) erase the number of temporary nodes
c) eliminate records where number of permanent nodes is 0
d) set new length for the block and rewrite it.

Module : 2.23 ITRO

Purpose : assign internal numbers to transit-only nodes



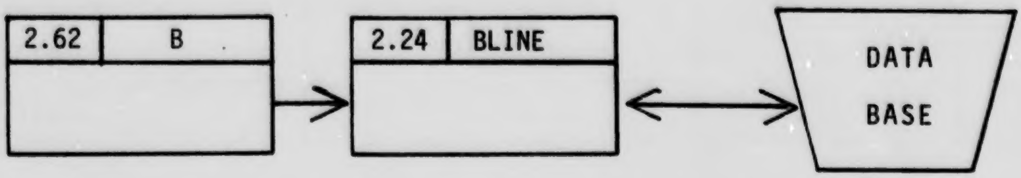
input:
1. file name: STOPS (file of nodes encountered on transit lines)
   type: data base
   contents: node numbers and frequency
2. vector of sorted centroid numbers
3. vector of sorted turning nodes
4. vector of regular road nodes

output: vector of transit-only nodes, sorted.

procedure:
a) reserve an area for transit-only nodes
b) read the file, record by record
c) determine which kind of node it is
d) when unknown, add it to the list
e) sort the list.

Module : 2.24 BLINE

Purpose : clean up file of transit line



input: file name: LINES (file of transit lines)

type: data base

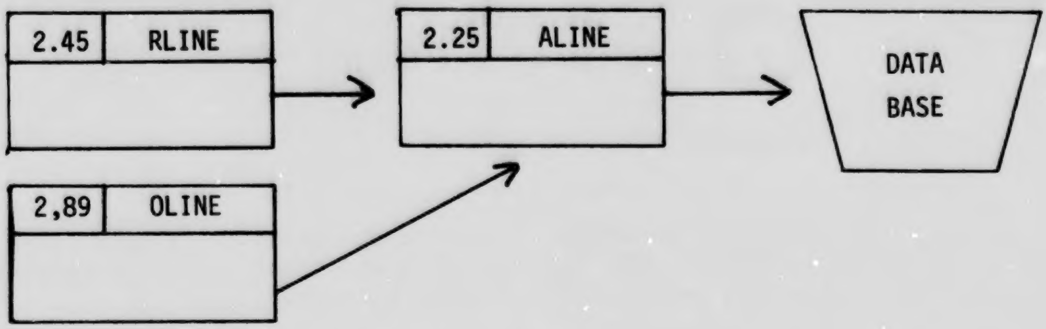
contents: data describing each transit line

output: same file as input

procedure:
a) read the file, record by record
b) transfer to another record all active lines or lines temporarily deactivated
c) erase the internal number of nodes
d) rewrite the clean records

Module : 2.25 ALINE

Purpose : add a transit line in data bank



input: parameters from RLINE and OLINE

LINENO, the number of the line

NAME, the name of the line (dimensioned to 20)

HDWAY, the headway

ALENF, length of the line

SPEED of the line

ITYPE

ALAOUR, the lay over time

IVICLE, the number of vehicles

MBDESC, the number of stops

IDESC, a vector containing alternately a stop and a direction

$$\text{IACT} \begin{cases} = 0 & \text{if permament correction} \\ = 1 & \text{if temporary modification} \end{cases}$$

output: 1. file name: LINES (file of transit lines)

type: data base

status: always modified

contents: data describing each transit line

sequence: sequential but also accessible randomly

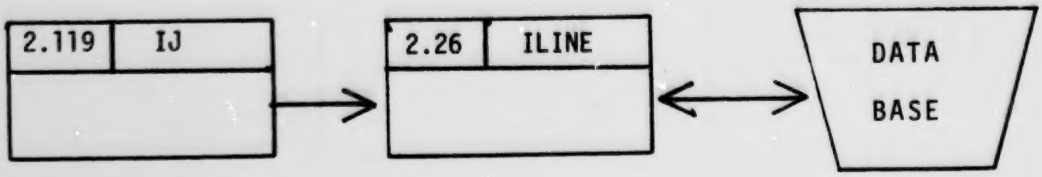
source: user specified

2. file name: STOPS (file of nodes encountered on transit lines)
   type: data base
   status: always modified
   contents: node numbers and frequence of apparition
   sequence: sequential
   source: created from file LINES

procedure: a) check if already defined
b) add the transit line at end of file
c)update the count of nodes encountered on transit lines; this is done for each node and count is kept separately for permanent and temporary nodes.

Module : 2.26 ILINE

Purpose : assign internal numbers to the nodes on transit lines



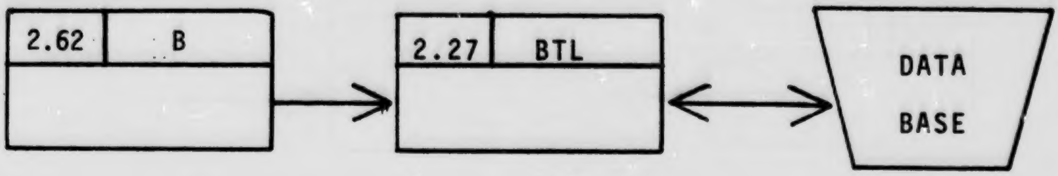
input:
1. file name: LINES (file of transit lines)
   type: data base
   contents: data describing each transit line
2. vector of sorted centroids numbers
3. vector of sorted turning nodes
4. vector of road-only nodes
5. vector of sorted transit-only nodes

output: same file as input

procedure:
a) read the file, record by record
b) for each node of the route, determine in which of the 4 vectors it is located. Only road-only and transit-only nodes are acceptable
c) multiply the node number by 2^{14} and add the internal number corresponding to the external number
d) rewrite the record on file.

Module : 2.27 BTL

Purpose : clean up file of transit-only links.



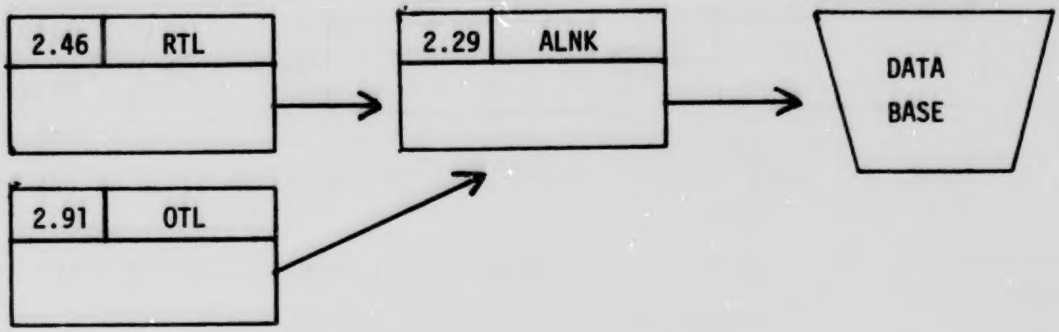
input: file name: ACCESS, EGRESS and TLTIM (file of transit-only links)
type: data base
contents: estimated travel times on transit links

output: same file as input

procedure:
a) repeat the following steps for each section of the file
b) read in a block
c) erase each record added temporarily or purged permanently; reactivate each record temporarily deactivated
d) separate second word of each record, restoring first and second word (see description of files)
e) set new length of block and rewrite it.

Module : 2.29 ALNK

Purpose : add user specified travel time for transit-only links in data bank.



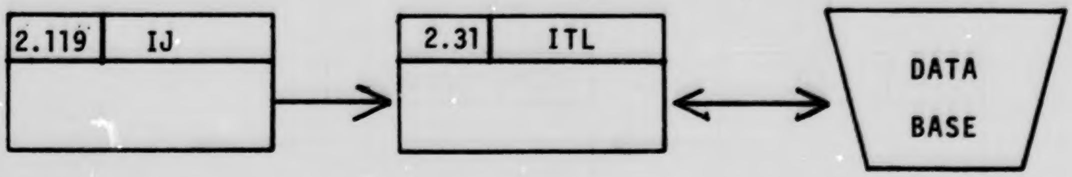
input:
parameters from RTL and OTL
origin node
destination node
travel time
IACT = 0 if permanent correction
= 1 if temporary modification

output:
file name: ACCESS, EGRESS or TLTIM (file of transit-only links)
type: data base
status: always modified
contents: estimated travel times on specified links
sequence: sequential in three sections
source: user specified

procedure:
a) determine if it is an access link, an egress link or a normal link
b) add it at the end of its section

Module ; 2.31 ITL

Purpose : assign internal numbers to nodes on transit links



input: 1. file name: ACCESS, EGRESS and TLTIM (file of transit-only links)
type: data base
contents: estimated travel times on transit links

2. vector of sorted centroid numbers

3. vector of sorted turning nodes

4. vector of road nodes

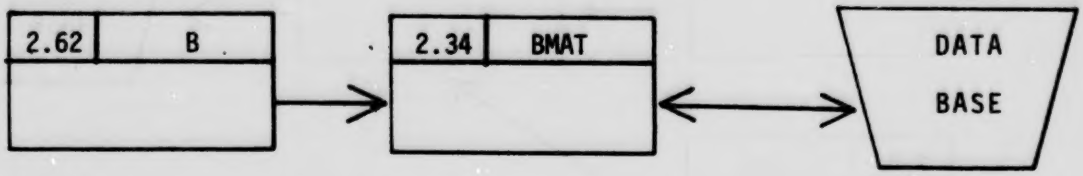
5. vector of sorted transit-only nodes

output: same files as input

procedure:
a) repeat the following steps for each section of the file
b) reserve an area for each link
c) read each record of the section
d) find which kind of node is origin and destination
e) pack, in the first word of the record, internal numbers of origin and destination; pack, in the second word, external numbers (see descriptions of files)
f) sort the records in core according to the first word of each. Use a bubblesort if a quicksort has already been used formerly; otherwise use a quicksort
g) put back sorted records on disk.

Module : 2.34 BMAT

Purpose : clean up all O-D matrices



input: the 6 following files:

O-D matrix # 1 ⎤
O-D matrix # 2 ⎥ PERSPQ
O-D matrix # 3 ⎦
vehicle adjustment matrix VEHADJ
auto occupancy AUTOCC
modal split MODEST

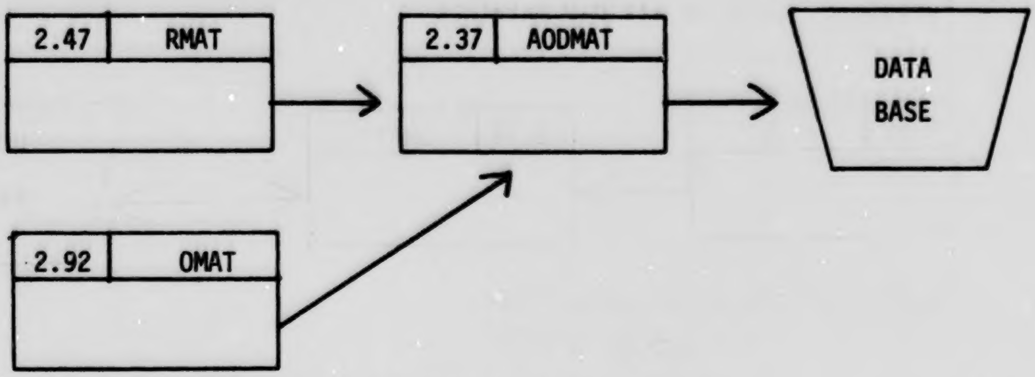
type: data base
contents: definition of a matrix

output: same files as input

procedure:
a) repeat the following steps for each matrix
b) read in the rows of the matrix, i.e. the list of origin nodes
c) keep only active rows and rows temporarily deactivated
d) collect any block disappearing in the process
e) read in, one by one, the columns of the matrix by means of the pointers associated with origins
f) keep only active columns and those temporarily deactivated
g) eliminate internal node numbers
h) rewrite the matrix in data bank.

Module : 2.37 AODMAT

Purpose : insertion of values in an O-D matrix in data bank



input:
parameters from RMAT and OMAT
identification of the matrix
origin node
number of destinations
vector of destination nodes
vector of values associated with destination nodes

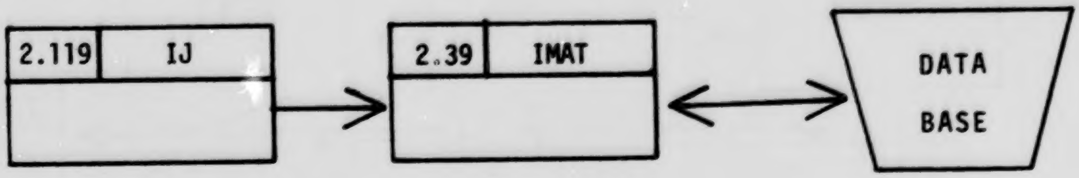
$$\text{IACT} \begin{cases} = 0 & \text{if permanent correction} \\ = 1 & \text{if temporary modification} \end{cases}$$

output:
one of the following files
    O-D matrix # 1
    O-D matrix # 2
    O-D matrix # 3
    vehicle adjustment matrix
    auto occupancy
    modal split
type: data base
contents: definition of a matrix
sequence: sequential with index
source: user specified

procedure:
a) find the row (origin node)
b) if the row is partly defined, get the section of defined columns
c) if the row is undefined, create a new section and enter the row with a pointer to the new section
d) add the new columns and their values (destination nodes)

Module : 2.39 IMAT

Purpose: assign internal numbers to centroids in all O-D matrices



input:

1. the 6 following files

   O-D matrix # 1
   O-D matrix # 2 PERSPQ
   O-D matrix # 3
   vehicle adjustment matrix VEHADJ
   auto occupancy AUTOCC
   modal split MODEST

2. the sorted list of centroids

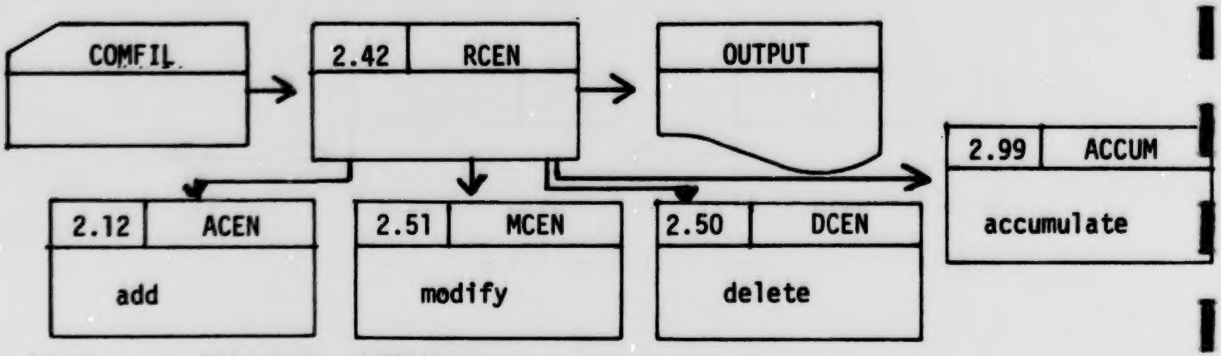
output: same files as input

procedure:

a) repeat the following steps for each matrix
b) read in the rows, i.e. the list of origin nodes
c) examine each node and find the internal number
d) multiply the external number by 2^{14} and add the internal number. Sort according to internal numbers
e) rewrite the rows in data bank
f) read in each columns, by means of the pointers associated with origin
g) find the external number of each destination
h) multiply the external number by 2^{14} and add the internal number
i) sort according to internal numbers
j) rewrite the rows in data bank

Module : 2.42 RCEN

Purpose : read centroids from information supplied by user


COMFIL
2.42
RCEN
OUTPUT
2.99
ACCUM
accumulate
2.12
ACEN
add
2.51
MCEN
modify
2.50
DCEN
delete

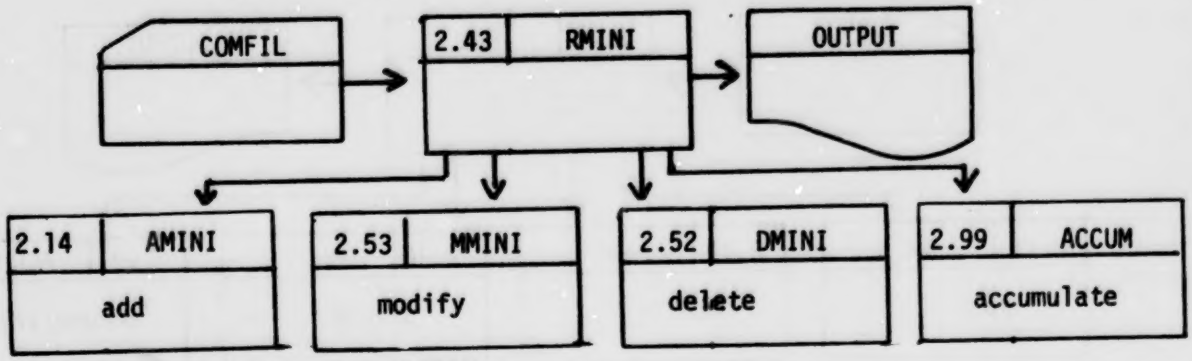
input : file name: COMFIL
type: card images
status: required
contents: see "centroid coding form"
sequence: sequential
source: user supplied

output: file name: OUTPUT
type: report file
status: will be routed to printer
contents: listing of input cards and error messages
sequence: sequential

procedure: a) read in card images
b) check validity of all fields
c) if no error and correction mode, call ACEN, MCEN or DCEN
d) if no error and scenario mode, call ACCUM.

Module : 2.43 RMINI

Purpose : read transit node - road nodes correspondence from information supplied by user.


COMFIL
2.43
RMINI
OUTPUT
2.14
AMINI
add
2.53
MMINI
modify
2.52
DMINI
delete
2.99
ACCUM
accumulate

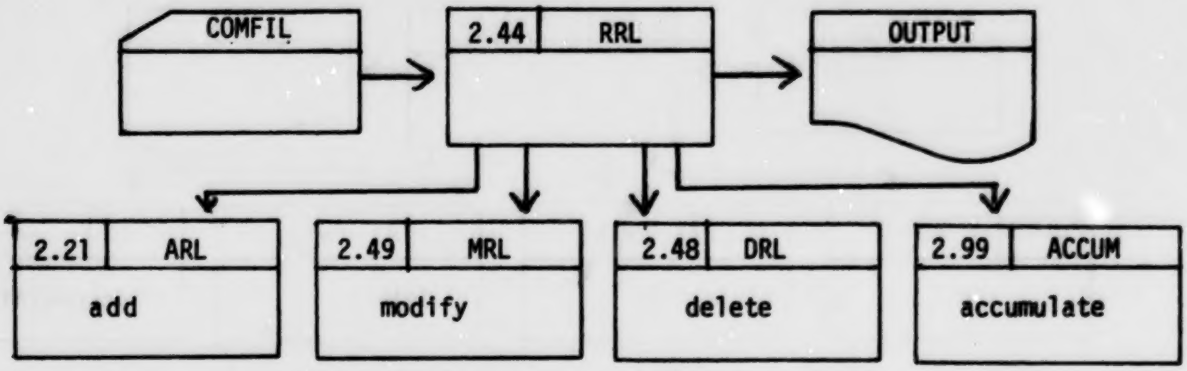
input :
file name: COMFIL
type: card images
status: required
contents: see coding form
sequence: sequential
source: user specified

output :
file name: OUTPUT
type: report file
status: will be routed to printer
contents: listing of input cards and error messages
sequence: sequential

procedure:
a) read in card images
b) check validity of all fields
c) in correction mode, call AMINI, MMINI or DMINI
d) in scenario mode, call ACCUM.

Module : 2.44 RRL

Purpose : read road links from information supplied by user


COMFIL
2.44
RRL
OUTPUT
2.21
ARL
add
2.49
MRL
modify
2.48
DRL
delete
2.99
ACCUM
accumulate

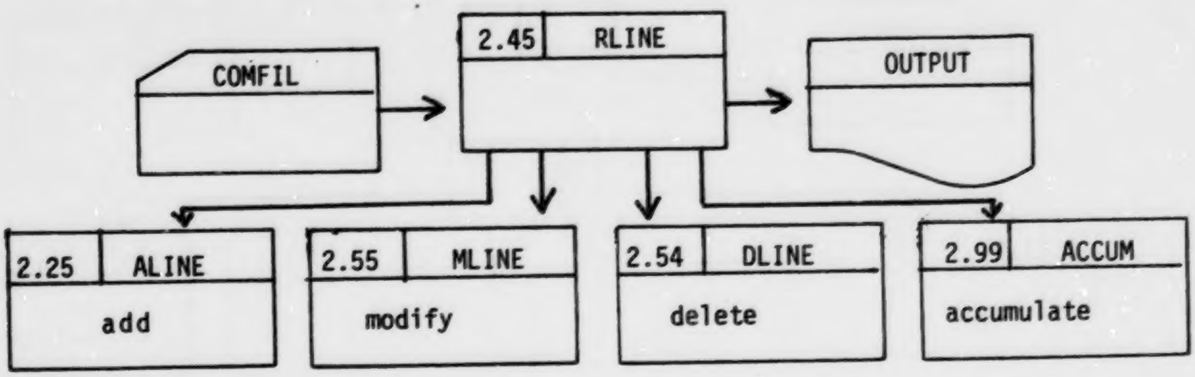
input: file name: COMFIL
type: card images
status: required
contents: see "road link coding form"
sequence: sequential
source: user specified

output: file name: OUTPUT
type: report file
status: will be routed to printer
contents: listing of input cards and error messages
sequence: sequential

procedure: a) read in card images
b) check validity of all fields
c) in correction mode, call ARL, MRL or DRL
d) in scenario mode, call ACCUM
e) in case of symmetry (correction mode), repeat step c).

Module ; 2.45 RLINE

Purpose : read transit lines from information supplied by user.


COMFIL
2.45
RLINE
OUTPUT
2.25
ALINE
add
2.55
MLINE
modify
2.54
DLINE
delete
2.99
ACCUM
accumulate

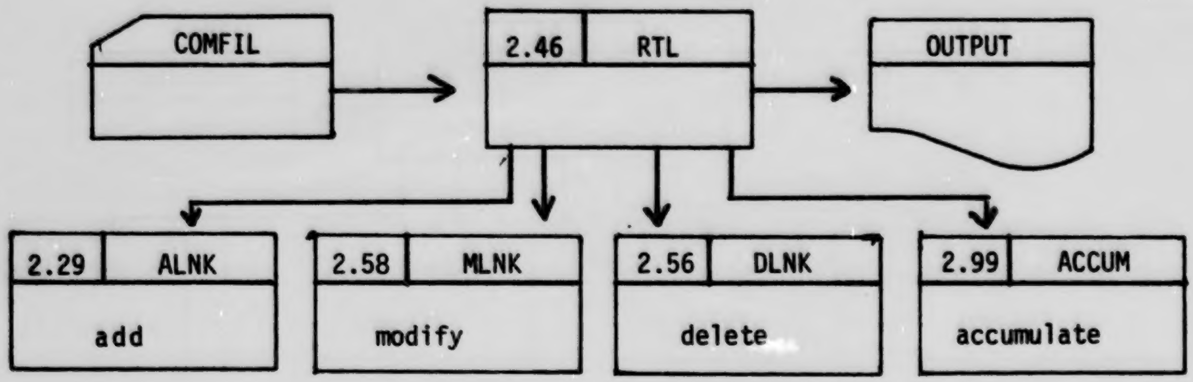
input : file name: COMFIL
type: card images
status: required
contents: see "transit lines coding form"
sequence: sequential
source: user specified

output : file name: OUTPUT
type: report file
status: will be routed to printer
contents: listing of input cards and error messages
sequence: sequential

procedure :
a) read in card images
b) accumulate characteristics of the transit line and stop numbers from the following cards
c) check validity
d) in correction mode, call ALINE, MLINE or DLINE
e) if in scenario mode, call ACCUM

Module : 2.46 RTL

Purpose : read transit links from information supplied by user


COMFIL
2.46
RTL
OUTPUT
2.29
ALNK
add
2.58
MLNK
modify
2.56
DLNK
delete
2.99
ACCUM
accumulate

input: 1. file name: COMFIL

type: card images

status: required

contents: see "transit link" and "transit access link" coding form

sequence: sequential

source: user specified

2. parameter from data base supervisor indicating "transit link" or "transit access links"

output: file name: OUTPUT

type: report file

status: will be routed to printer

contents: listing of input cards and error messages

sequence: sequential

procedure: a) read in card images

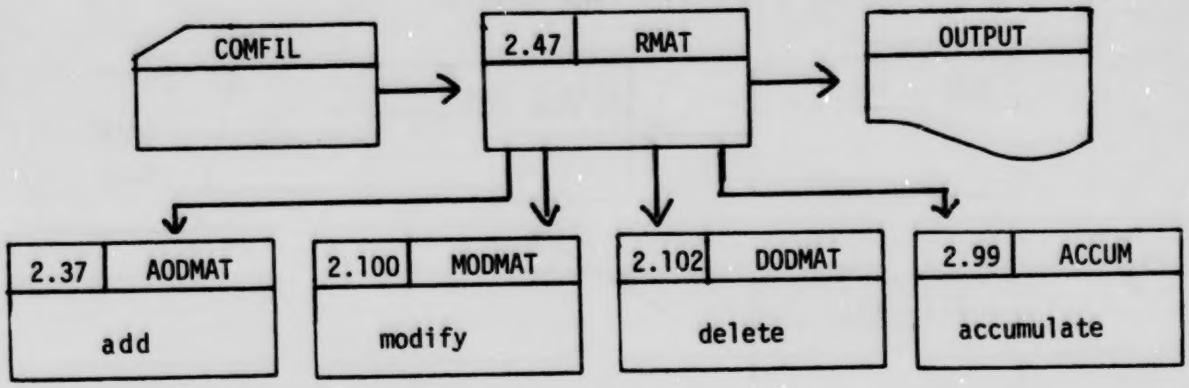
b) check validity of all fields

c) in scenario mode, call ACCUM

d) in correction mode, call ALNK, MLNK or DLNK either once or twice depending on reversability of the links

Module : 2.47 RMAT

Purpose: read a matrix from information supplied by user



COMFIL
2.47
RMAT
OUTPUT
2.37
AODMAT
add
2.100
MODMAT
modify
2.102
DODMAT
delete
2.99
ACCUM
accumulate

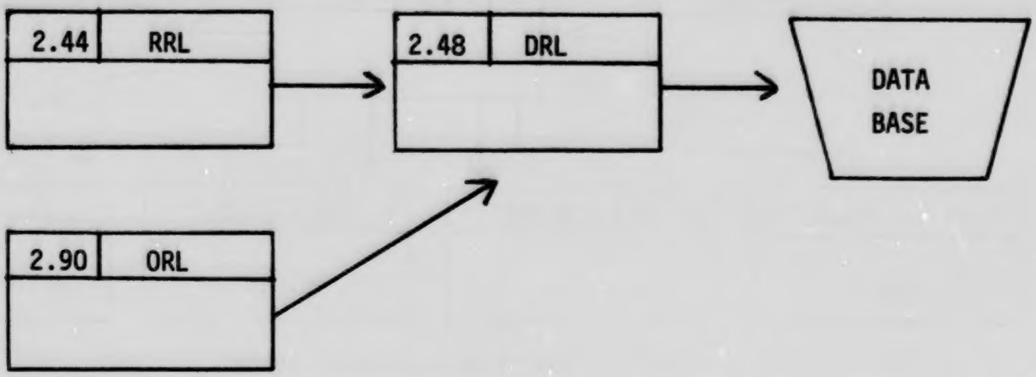
input:
1. parameter from data base supervisor
   =1 for O-D matrix # 1
   =2 for O-D matrix # 2
   =3 for O-D matrix # 3
   =4 for auto occupancy matrix
   =5 for modal split matrix
   =23 for vehicle adjustment matrix
2. file name: COMFIL
   type: card images
   status: required
   contents: see "O-D matrix coding form"
   source: user specified

output:
file name: OUTPUT
type: report file
status: will be routed to printer
contents: listing of input cards and error messages
sequence: sequential

procedure:
a) read in card images
b) accumulate destinations and values
c) check validity of all fields
d) in correction mode, call AODMAT, MODMAT or DODMAT
e) in scenario mode, call ACCUM.

Module : 2.48 DRL

Purpose : delete a road link from data bank



input:

1. parameters from RRL and ORL

   origin node

   destination node

   IACT { = -2 if permanent correction

   = -1 if temporary modification

2. The list of known road links and the position of the records on file, prepared by BRL

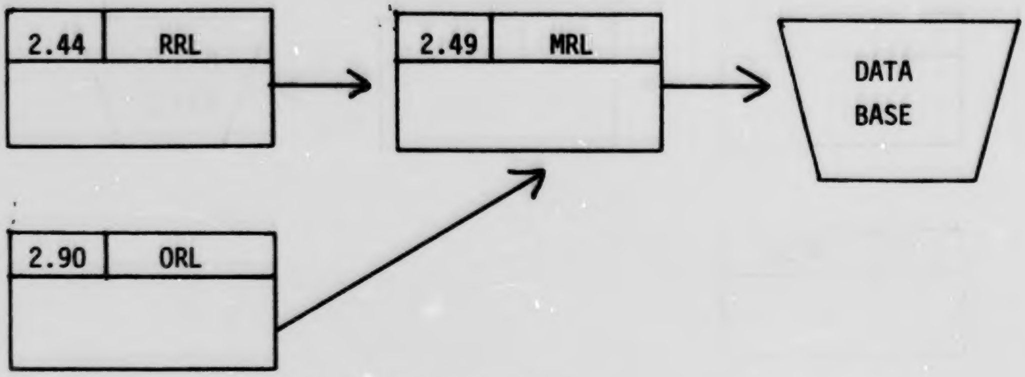
output: see module 2.21 ARL

procedure:

a) locate the link on file, using the list
b) mark the record as no longer valid
c) if permanent correction, cancel the entry of the record in the list.

Module : 2.49 MRL

Purpose : modify a road link in data bank



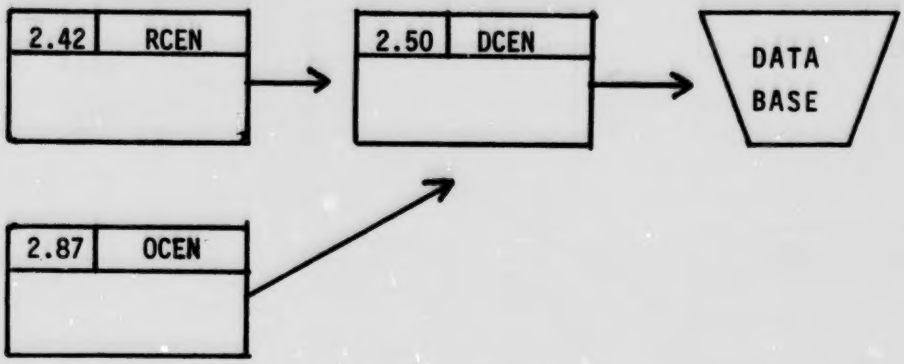
input:
1. parameters from RRL and ORL
   origin node
   destination node
   new type, if any
   new length, if any
   new observed time, if any
   new observed volume, if any
   5 new user defined data, if any
   new text, if any (16 characters)
   IACT = 0 if permanent correction
   IACT = 1 if temporary modification
2. The list of known road links and the position of the records on file, prepared by BRL

output: see module 2.21 ARL

procedure:
a) locate the link on file, using the list
b) replace values in the record if permanent correction to permanent link or temporary modification to temporary link
c) in other cases, mark the record as inactive, add a new one in the file and update pointer in the list.

Module : 2.50 DCEN

Purpose : delete a centroid from data bank


2.42
RCEN
2.50
DCEN
DATA
BASE
2.87
OCEN

input : Parameters from RCEN and OCEN

NUMBER of centroid

IACT = -2 if permanent correction
= -1 if temporary modification

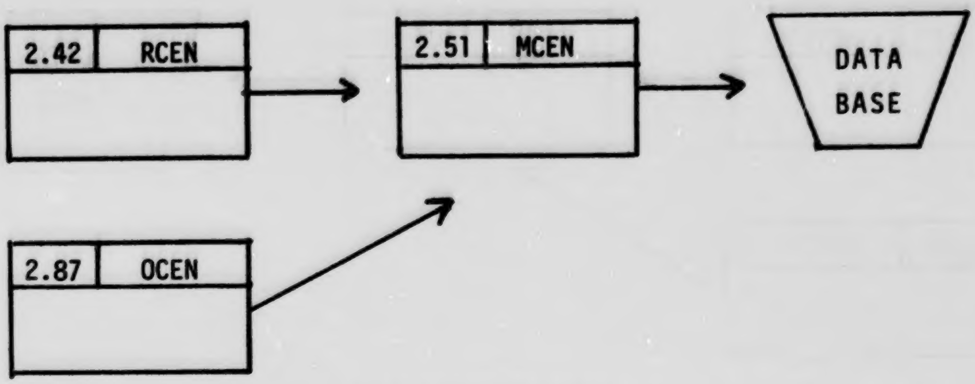
output : see module 2.12 ACEN

procedure : a) check if defined

b) mark the centroid as no longer valid, according to IACT

Module : 2.51 MCEN

Purpose : Modify a centroid in data



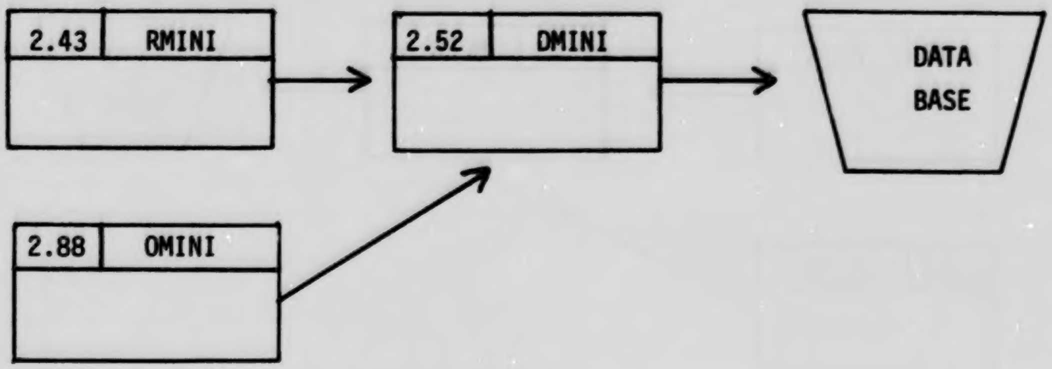
input : Parameters from RCEN and OCEN
NUMBER of the centroid
new X - coordinate
MX = true if X - coordinate is to be changed
new Y - coordinate
MY = true if Y - coordinate is to be changed
IACT = 0 if permanent correction
IACT = 1 if temporary modification

output : see module 2.12 ACEN

procedure :
a) check if defined
b) if IACT = 0 change X, Y and NAME in the record
c) if IACT = 1 mark the record as temporarily inactive and create a temporary record at the end of file.

Module : 2.52 DMINI

Purpose : delete an entry in the correspondence table in data bank.



input : parameters from RMINI and OMINI

NODE at center of expansion

IACT $\left\{ \begin{array}{l} = -2 \text{ if permanent correction} \\ = -1 \text{ if temporary modification} \end{array} \right.$

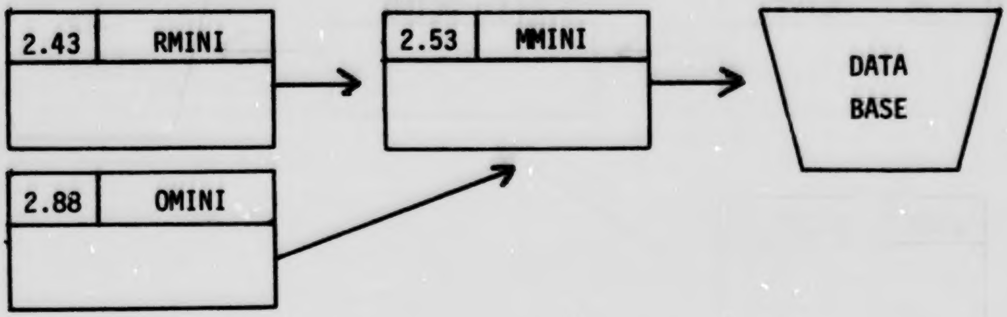
output : see module 2.14 AMINI

procedure : a) find the expansion in the file

b) mark it as no longer valid, according to IACT.

Module : 2.53 MMINI

Purpose : modify a node expansion in data bank.



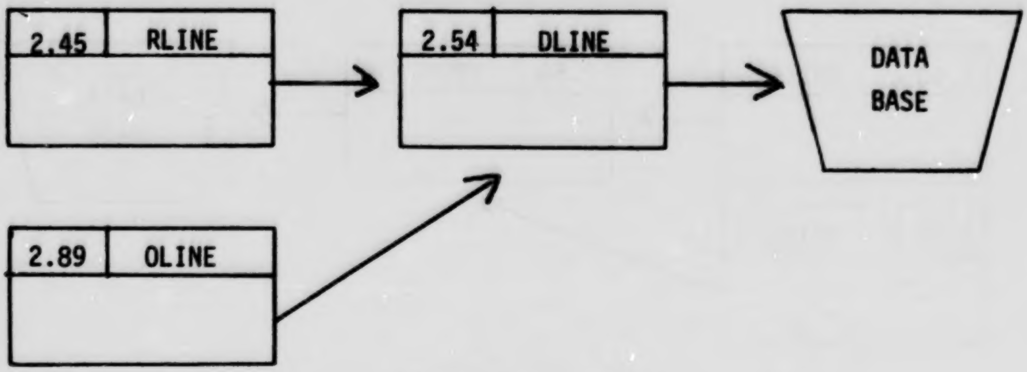
input : parameters from RMINI and OMINI
NODE at center of expansion
NB, the new number of nodes in the expansion
LIST, vector containing new nodes in the expansion
IACT $\begin{cases} = 0 & \text{if permanent correction} \\ = 1 & \text{if temporary modification} \end{cases}$

output : see module 2.14 AMINI

procedure : a) perform a DMINI using NODE and IACT-2
b) perform a AMINI using all arguments

Module : 2.54 DLINE

Purpose : delete a transit line from data bank



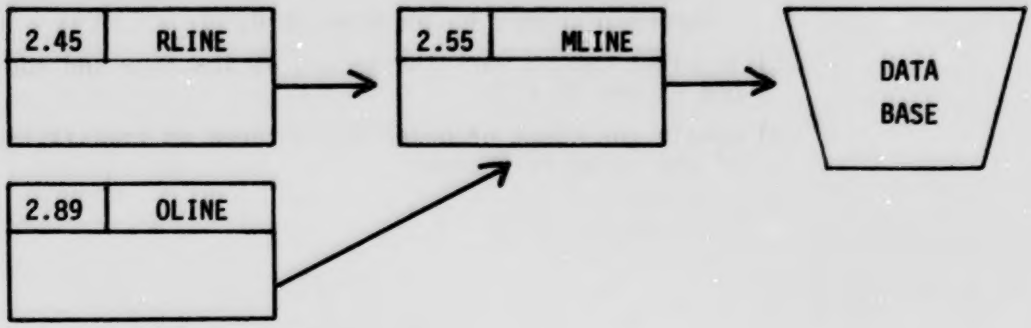
input: parameters from RLINE and OLINE
line number
name of transit line (dimensioned to 20)
IACT $\begin{cases} = -2 & \text{if permanent correction} \\ = -1 & \text{if temporary modification} \end{cases}$

output: see module 2.25 ALINE

procedure: a) find the transit line by number and name
b) mark the transit line as no more valid
c) update the file of nodes encountered on transit line using the old bus stops in the record.

Module : 2.55 MLINE

Purpose: modify a transit line in data bank



input:
parameters from RLINE and OLINE
the line number
the new name of the line (dimensioned to 20)
the new headway, if any
the new length, if any
the new speed, if any
the new type, if any
the new lay over time, if any
the new number of vehicles, if any
the new number of stops
the new stop numbers and directions

IACT $\begin{cases} = 0 & \text{if permanent correction} \\ = 1 & \text{if temporary modification} \end{cases}$

output:

1. file name: LINES
   see module 2.25 ALINE
2. file name: STOPS (file of nodes encountered on transit lines)
   see module 2.25 ALINE

procedure: a) check if transit line is defined

b) make correction in the record if these conditions are satisfied

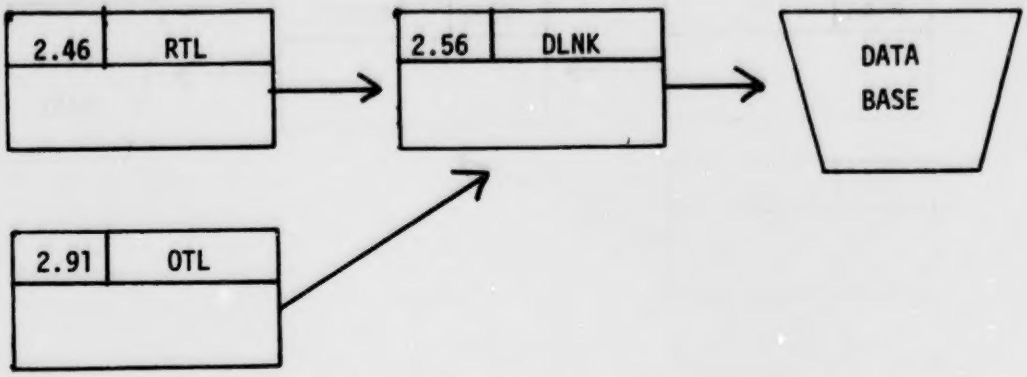
  - permanent correction to permanent line or temporary modification to temporary line
  - number of stops is the same or 0.

c) in other cases, mark the record as inactive and add a new one at end of file

d) update the count of nodes encountered on transit lines if the route is changed.

Module : 2.56 DLNK

Purpose : delete a transit-only link from data bank



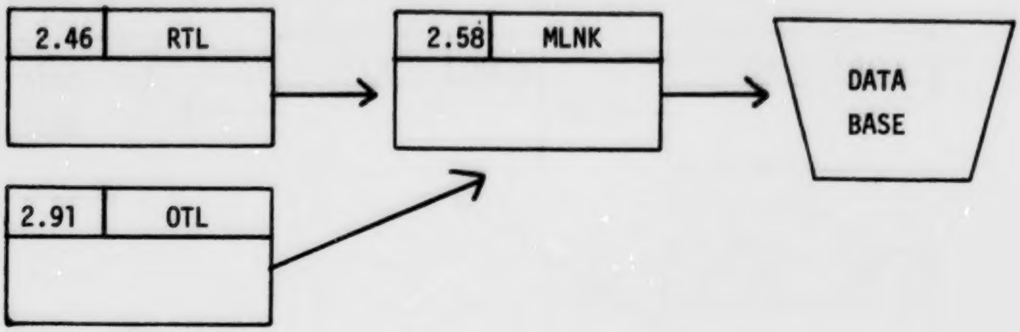
input: parameters from RTL and OTL
origin node
destination node
IACT $\begin{cases} = -2 & \text{if permanent correction} \\ = -1 & \text{if temporary modification} \end{cases}$

output: see module 2.29 ALNK

procedure: a) determine in which section of the file, the link is located
b) mark it as no longer valid

Module: 2.58 MLNK

Purpose : modify travel time for transit-only links in data bank.



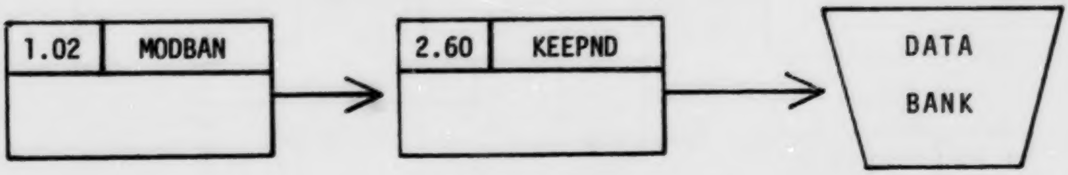
input: parameters from RTL and OTL
origin node
destination node
new travel time
IACT = 0 if permanent correction
= 1 if temporary modification

output: see module 2.29 ALNK

procedure:
a) determine in which section of the file, the link is located
b) change value if permanent correction of permanent link or temporary modification of temporary link
c) otherwise mark the link as inactive and add a new one at the end of the section.

Module : 2.60 KEEPND

Purpose : keep node numbers in databank



input:

1. sorted vector of centroid numbers
2. sorted vector of turning nodes
3. vector of regular road nodes
4. sorted vector of transit-only nodes

output:

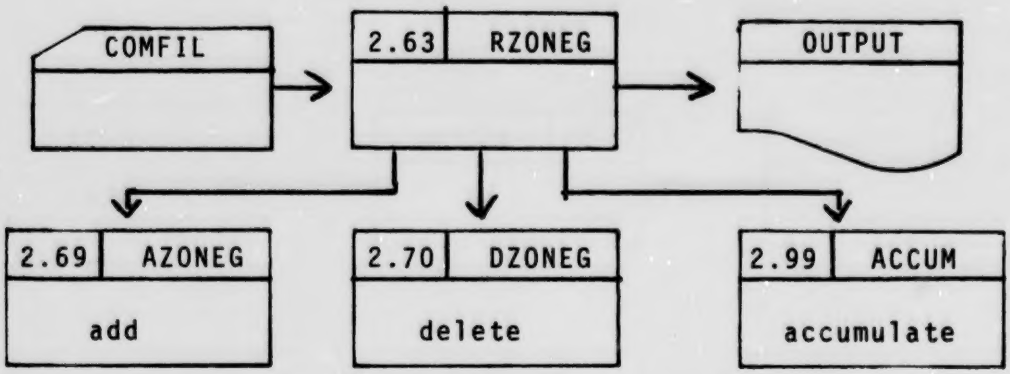
file name: NODEXT (file of internal-external numbers)
type: data base
position in index: 54

procedure:

a) eliminate the output file, if one exists
b) copy the vector of centroid numbers as first record
c) copy the vector of turning nodes as second record
d) copy the vector of regular road nodes as third record
e) copy the vector of transit-only nodes as fourth record.

Module : 2.63 RZONEG

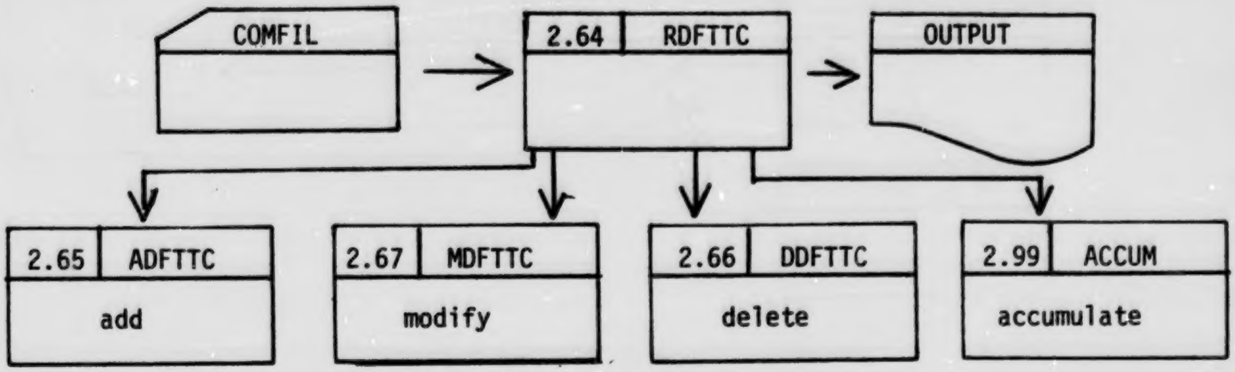
Purpose : read zone grouping from information supplied by user


COMFIL
2.63
RZONEG
OUTPUT
2.69
AZONEG
add
2.70
DZONEG
delete
2.99
ACCUM
accumulate

input:	file name: COMFIL
	type: card images
	status: required
	contents: see "zone group coding form"
	sequence: sequential
	source: user specified
output:	file name: OUTPUT
	type: report file
	status: will be routed to printer
	contents: listing of input cards and error messages
	sequence: sequential
procedure:	a) read in card images
	b) accumulate zones forming a group
	c) in correction mode, call AZONEG or DZONEG
	d) in scenario mode, call ACCUM

Module: 2.64 RDFTTC

Purpose: read demand function travel time coefficients from information supplied by user


COMFIL
2.64
RDFTTC
OUTPUT
2.65
ADFTTC
add
2.67
MDFTTC
modify
2.66
DDFTTC
delete
2.99
ACCUM
accumulate

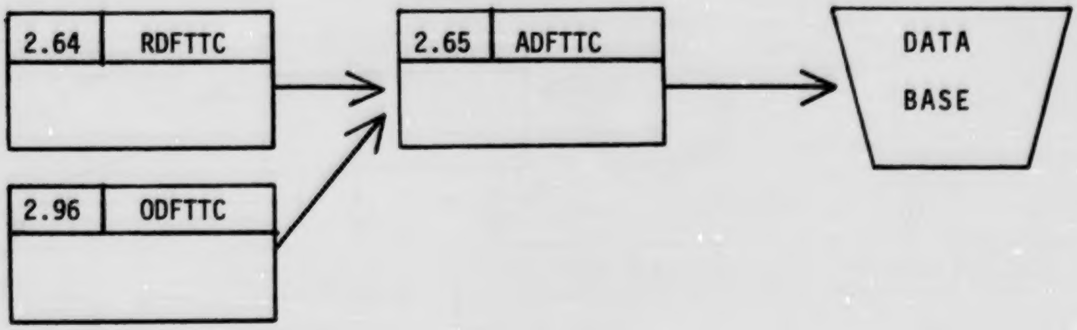
input: file name: COMFIL
type: card images
status: required
contents: see "demand function travel time coefficient coding form"
sequence: sequential
source: user specified

output: file name: OUTPUT
type: report file
status: will be routed to printer
contents: listing of input cards and error messages
sequence: sequential

procedure:
a) read in card images
b) get zone number and accumulate 3 coefficients
c) check validity of all fields
d) in modification mode, call ADFTTC, MDFTTC or DDFTTC
e) in scenario mode, call ACCUM

Module : 2.65 ADFTTC

Purpose: add demand function travel time coefficients to data bank



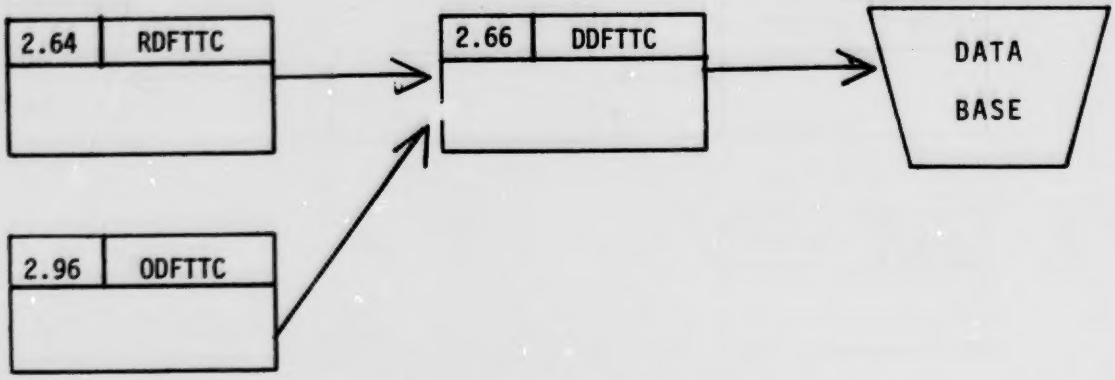
input:
parameters from RDFTTC and ODFTTC
name of zone, in characters
IACT = 0 if permanent correction
= 1 if temporary modification
3 coefficients

output:
file name: DCOKIN (file of demand function travel time coefficients)
type: data base
contents: coefficients associated with zones or group of zones
sequence: sequential
source: user specified

procedure:
a) check if already defined
b) add new coefficients

Module : 2.66 DDFTTC

Purpose : delete demand function travel time coefficients from data bank.


2.64
RDFTTC
2.66
DDFTTC
DATA
BASE
2.96
ODFTTC

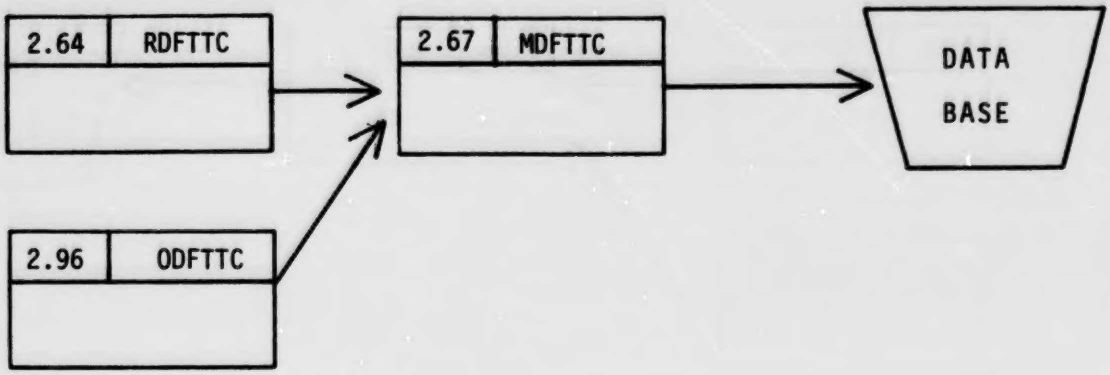
input: parameters from RDFTTC and ODFTTC
name of group, in characters
active indicator (permanent / temporary)

output: see module 2.65 ADFTTC

procedure: a) find the group or zone in the file
b) mark it as inactive

Module : 2.67 MDFTTC

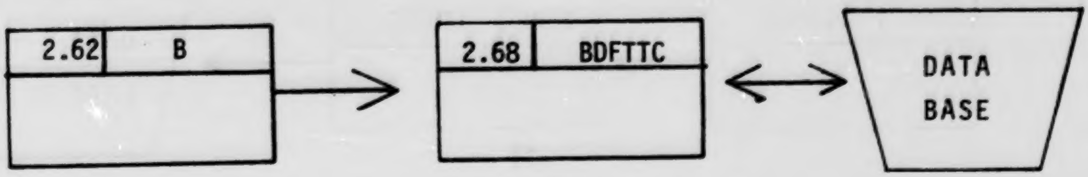
Purpose : modify demand function travel time coefficients in data bank



input:	parameters from RDFTTC and ODFTTC name of zone, in characters active indicator (permanent / temporary) 3 coefficients 3 indicators set to true if corresponding coefficients are to be modified
output:	see module 2.65 ADFTTC
procedure:	a) find the zone in the file b) if possible, change the coefficients in the record c) if not, mark the record as inactive and add a new one at the end of file.

Module : 2.68 BDFTTC

Purpose: clean up file of demand function travel time coefficients



input: 1. file name: DCOKIN (file name of demand function travel time coef
type: data base
contents: coefficients associated with zones or group of zones
source: user specified

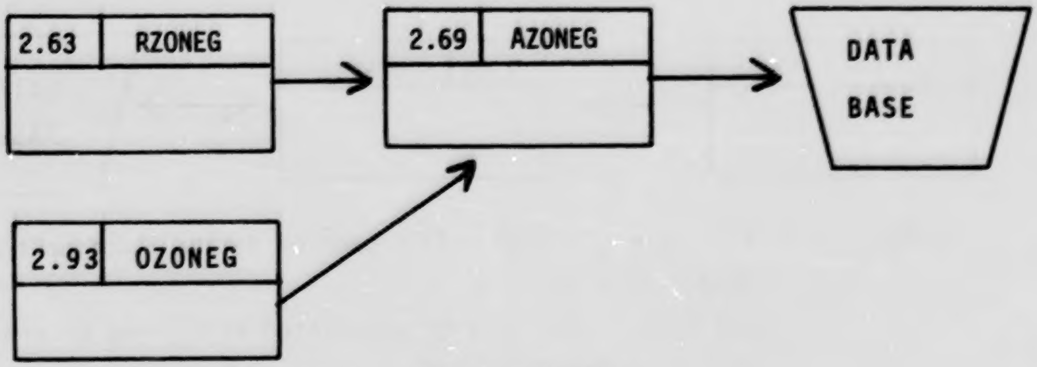
2. file name: DCOK (file of demand function travel time coeff.)
type: data base
contents: a vector of coefficients
source: constructed by IDFTTC

output: the first input file DCOKIN

procedure: a) eliminate file DCOK
b) read DCOKIN, record by record
c) eliminate temporary data and obsolete data
d) rewrite the clean records

Module : 2.69 AZONEG

Purpose : add a group of zones to data bank



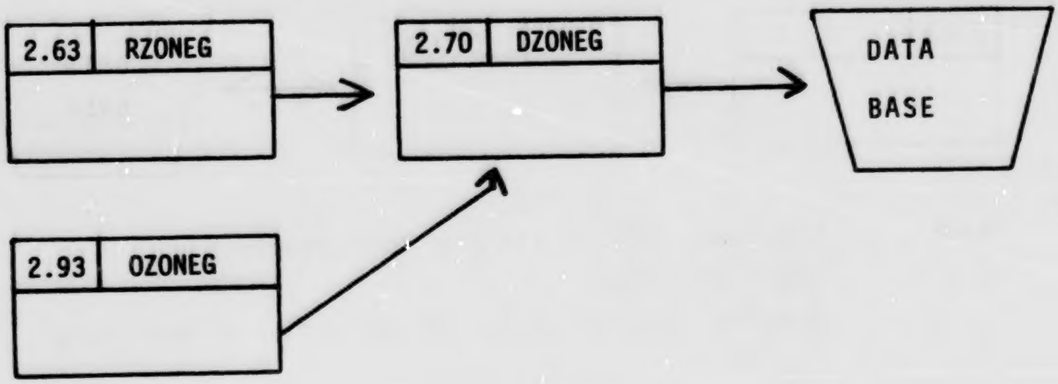
input: parameters from RZONEG and OZONEG
name of group
number of zones in the list
list of zones
IACT = 0 if permanent correction
= 1 if temporary modification

output: file name: ZGROUP (file of zone grouping)
type: data base
contents: name of groups and definition of each group
sequence: sequential with index
source: user specified

procedure:
a) check if the group is defined
b) if it is defined, get the list of zones and add the new partial list
c) if it is not defined, create the list, enter the name with pointer to the new list

Module : 2.70 DZONEG

Purpose : delete a (part of) group of zones from data bank



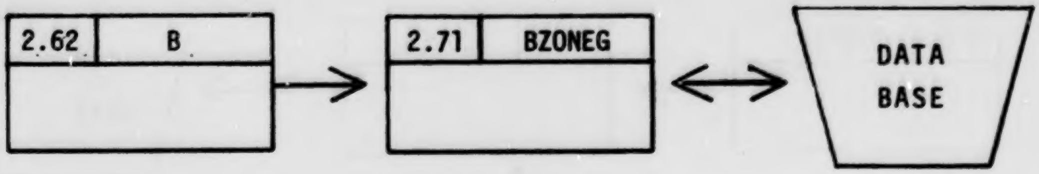
input:
parameters from RZONEG and OZONEG
name of the group
number of zones in the list
list of zones to be deleted
IACT = -2 if permanent correction
= -1 if temporary modification

output: see module 2.69 AZONEG

procedure:
a) check if the group is defined
b) if the list of zones to be deleted is empty, delete the whole group
c) otherwise mark as inactive the elements of the list.

Module : 2.71 BZONEG

Purpose : clean up file of zone grouping



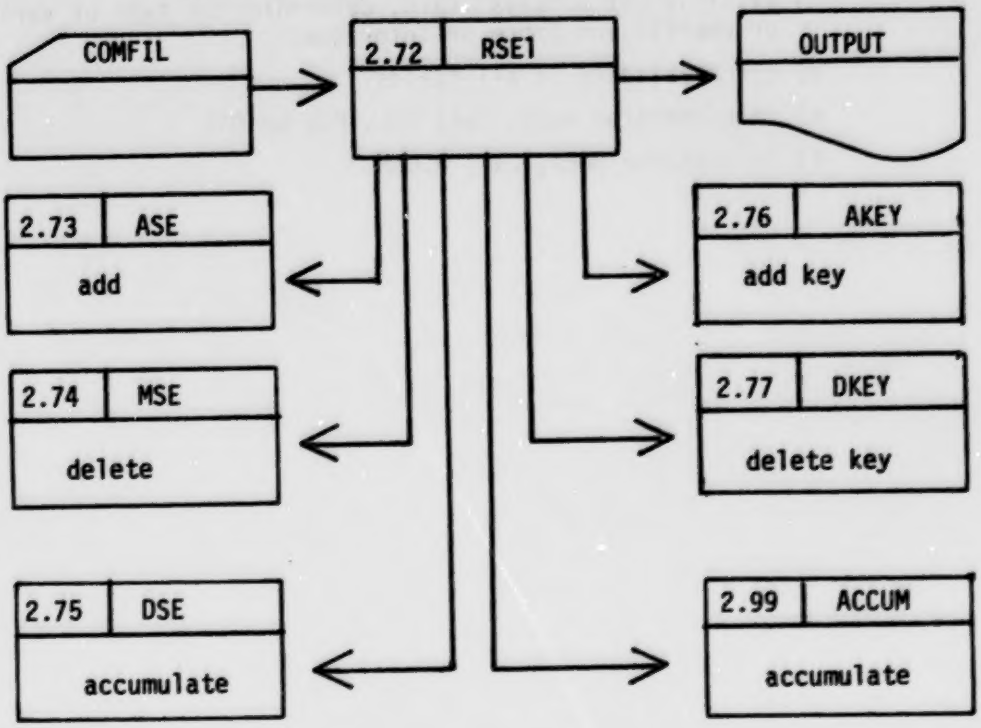
input: file name: ZGROUP (file of zone grouping)
type: data base
contents: name of groups and definition of each group

output: same file as input

procedure:
a) read first section of the file (a kind of index)
b) eliminate temporary and obsolete data
c) consider each remaining record
d) read in the definition of the group
e) merge the sequences of definition in a single list
f) rewrite the records in data bank
g) collect any unused block.

Module : 2.72 RSE1

Purpose: read socio-economic variables and coefficients from information supplied by user on coding form 1.


COMFIL
2.72
RSE1
OUTPUT
2.73
ASE
add
2.74
MSE
delete
2.75
DSE
accumulate
2.76
AKEY
add key
2.77
DKEY
delete key
2.99
ACCUM
accumulate

input:

1. parameter from data base supervisor
   = 0 variables
   = 1 coefficients
2. file name: COMFIL
   type: card images
   status: required
   contents: see "socio-economic variable / demand function coefficient coding form 1"
   sequence: sequential
   source: user specified

output:

file name: OUTPUT
type: report file
status: will be routed to printer
contents: listing of input cards and error messages
sequence: sequential

procedure: a) read in card images

b) if it is a "keys" card, get the zone or interzone keys and call AKEY or DKEY

c) if it is not a "keys" card, determine the type of variable or coefficient (zone or interzone)

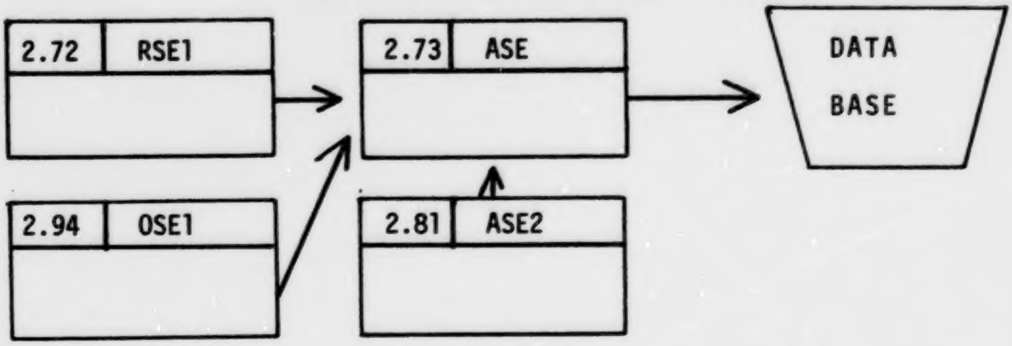
d) check validity of all fields

e) in correction mode, call ASE, MSE or DSE

f) in scenario mode, call ACCUM.

Module : 2.73 ASE

Purpose: add-socio economic variables and coefficients to data bank from coding form 1.



input: parameters from RSE1, OSE1 And ASE2

IVC = 0 variables
= 1 coefficients

name 1: origin group (4 characters)

name 2: destination group or empty (4 characters)

origin type = 1 origin is centroid
= 2 origin is group
= 3 origin is "all"

destination type = 0 no destination
= 1, 2, 3 as origin group

values of coefficients or variables (dimensioned to 8)

IACT = 0 if permanent correction
= 1 if temporary modification

output: file name: SOCIO (file of socio-economic data)

type: data base

contents: socio-economic data

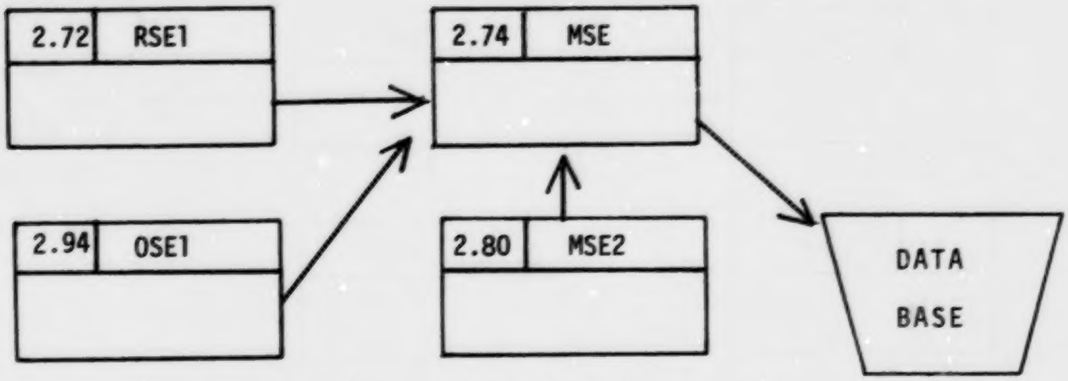
sequence: sequential with index

source: user specified

procedure:
a) check if already defined
b) if type is "all" or "all-all", place the values in the index of the file
c) in other cases add the value at end of file
d) repeat steps b and c for all keys.

Module : 2.74 MSE

Purpose : modify socio-economic variables and coefficients in data bank from coding form 1



input: parameters from RSE1, OSE1, MSE2

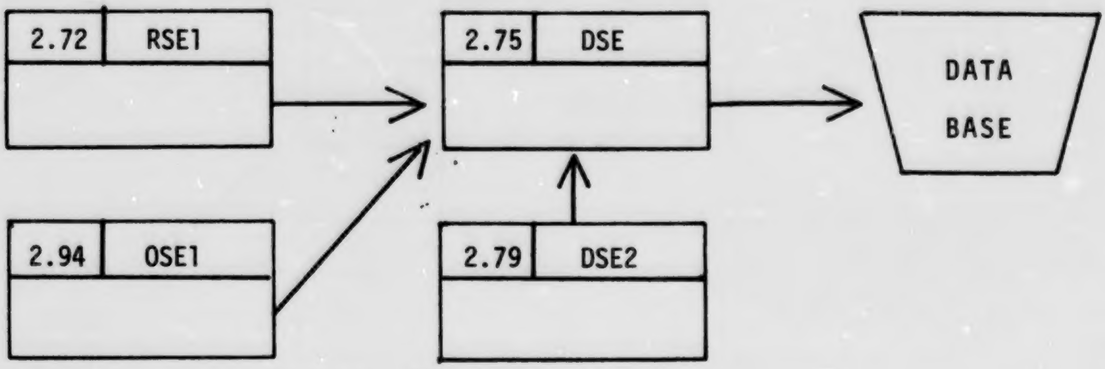
they are the same parameters as module 2.73 ASE with one more, an indicator which tells if each value is to be modified (dimensioned to 8)

output: see module 2.73 ASE

procedure:
a) check if defined
b) for all undefined values,add the new one
c) for those defined values,change it in the record if permanent correction to permanent values or temporary modification to temporary values
d) otherwise mark the value as inactive and add the new one at end of the file.
e) repeat steps b, c and d for each key

Module : 2.75 DSE

Purpose : delete socio-economic variables and coefficients in data bank from coding form 1.



input: parameters for RSE1, OSE1 and DSE2

IVC = 0 variables

= 1 coefficients

name of origin group

name of destination group (or empty)

origin type

destination type

IACT = -2 if permanent correction

= -1 if temporary modification

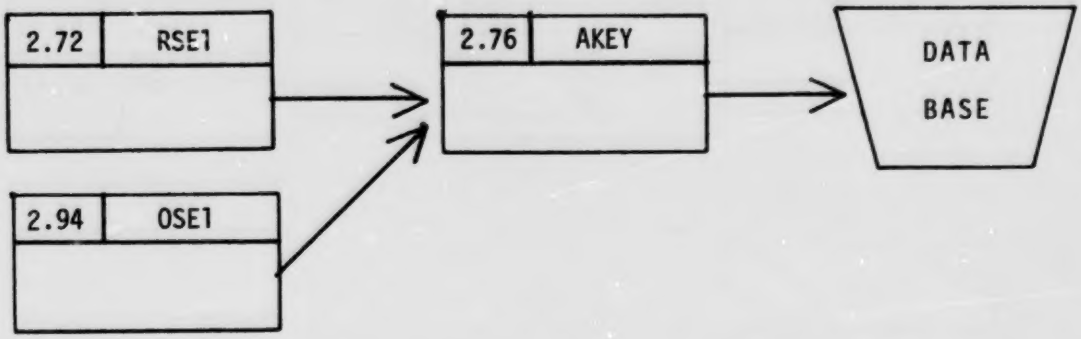
output: see module 2.73 ASE

procedure: a) if type is "all" or "all-all", cancel the value in the index of file.

b) in other cases, for each key, find the definition and mark it as inactive.

Module : 2.76 AKEY

Purpose : add keys of socio-economic variables and coefficients to data bank



input: parameters from RSE1 and OSE1

IVC = 0 variables

= 1 coefficients

type = 4 zone keys

= 5 interzone keys

a vector containing key names (dimensioned to 8)

number of keys transmitted

output: see module 2.73 ASE

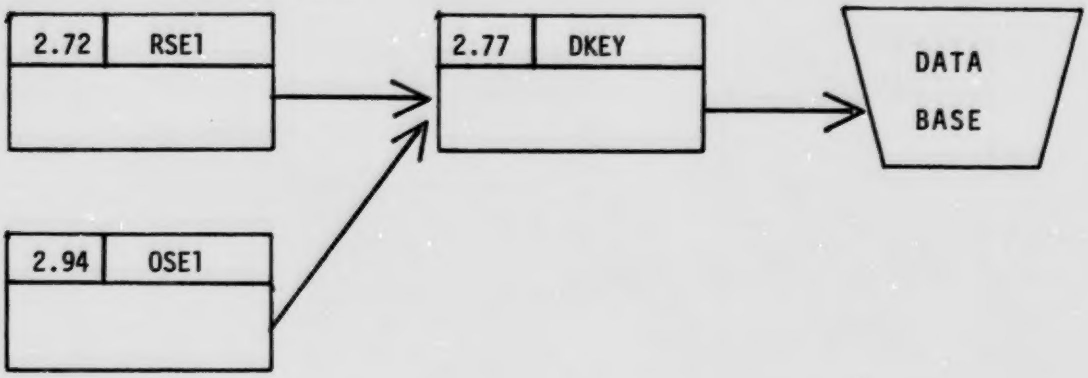
procedure: a) place the key name in the index of the output file

b) repeat for each key

note: the action is always permanent regardless of appearance of such data in a scenario.

Module : 2.77 DKEY

Purpose: delete keys of socio-economic variables and coefficients from data bank


2.72
RSE1
2.94
OSE1
2.77
DKEY
DATA
BASE

input: see module 2.76 AKEY

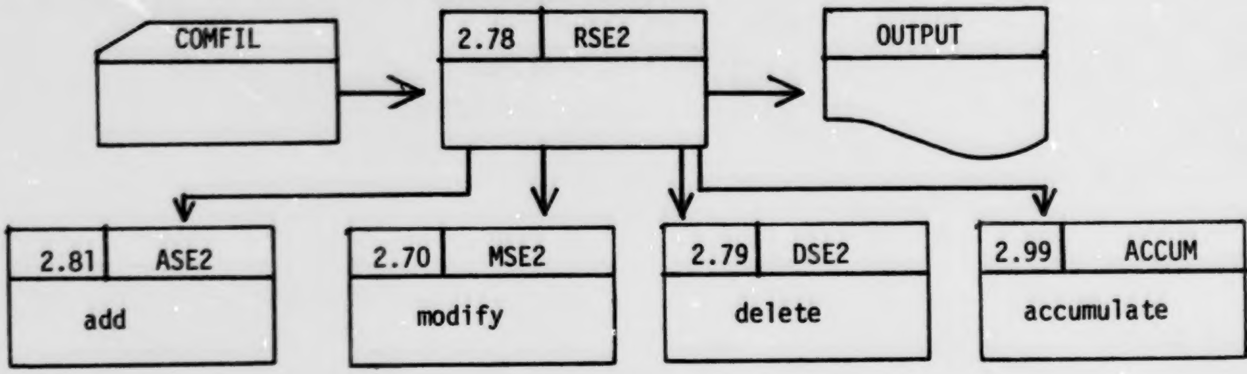
output: see module 2.73 ASE

procedure: a) find the key in the index of the file and erase it
b) repeat for each key transmitted

note: the action is always permanent.

Module : 2.78 RSE2

Purpose: read socio-economic variables and coefficients from information supplied by user on coding form 2.


COMFIL
2.78
RSE2
OUTPUT
2.81
ASE2
add
2.70
MSE2
modify
2.79
DSE2
delete
2.99
ACCUM
accumulate

input:

1. parameter from data base supervison
   = 0 variables
   = 1 coefficients

2. file name: COMFIL
   type: card images
   status: required
   contents: see "socio-economic variable / demand function coefficient coding form 2"
   sequence: sequential
   source: user specified

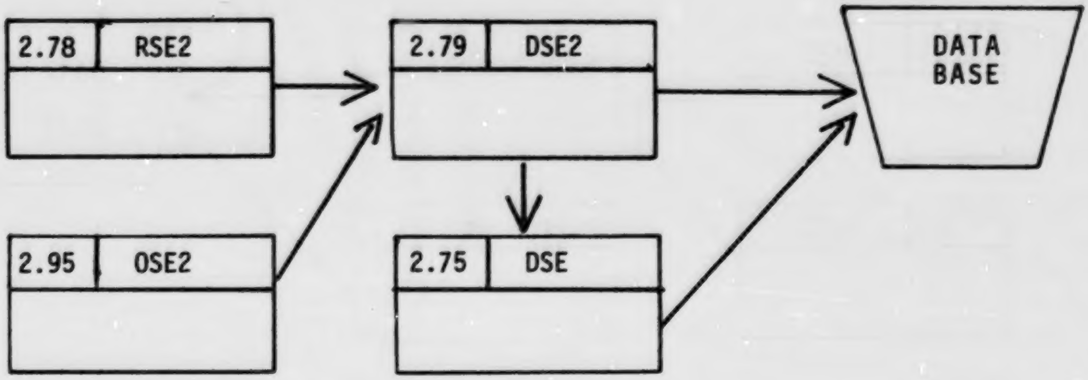
output:

file name: OUTPUT
type: report file
status: will be routed to printer
contents: listing of input cards and error messages
sequence: sequential

procedure:
- a) read in card images
- b) determine if the data are entered by zone or by key name
- c) when they are entered by zone accumulate key names and values. Check also the presence of a destination zone
- d) when they are entered by key name, find the origin zone and its value or accumulate the list of O-D pairs with their values
- e) in modification mode, call ASE2, MSE2 or DSE2
- f) in scenario mode, call ACCUM

Module : 2.79 DSE2

Purpose: delete socio-economic variables and coefficients in data bank from coding form 1.



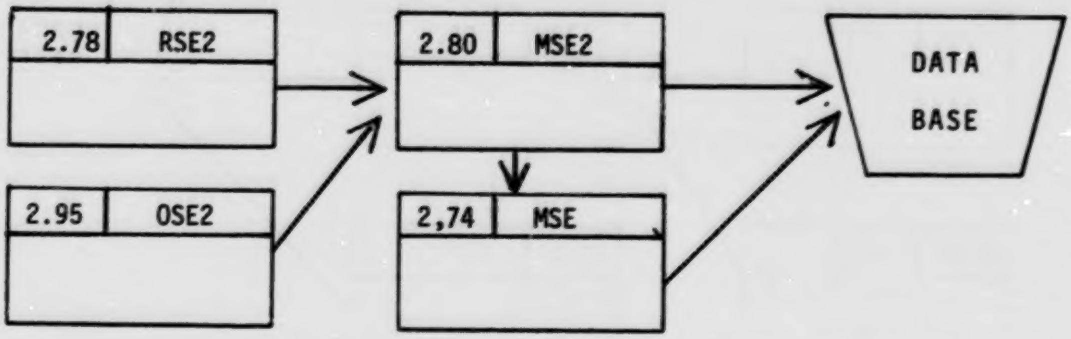
input: see module 2.81 ASE2 except
IACT = -2 if permanent correction
= -1 if temporary modification
and no vector of values is transmitted

output: see module 2.73 ASE

procedure:
a) if type ≠ 4 reorganize the parameters and call DSE
b) if type = 4 locate the key in the index
c) find origin and destination
d) mark the value as inactive
e) repeat steps b, c and d for each O-D pair.

Module : 2.80 MSE2

Purpose: modify socio-economic variables and coefficients in data bank from coding form 2.



input: see module 2.81 ASE2

output: see module 2.73 ASE

procedure:

a) if type ≠ 4, reorganize the parameters and call MSE to perform the job

b) if type = 4, locate the key in the index

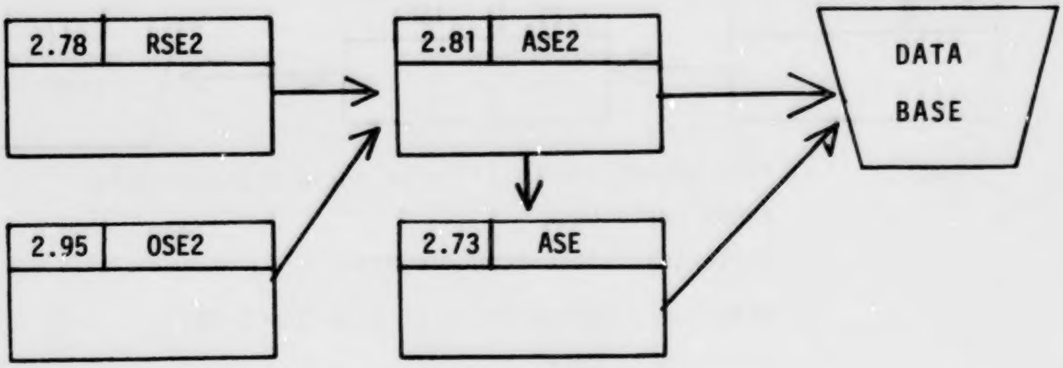
c) find origin and destination

d) change value in the record if possible; if not, mark the value as inactive and add a new one at end of record

e) repeat steps b, c and d for each O-D pair.

Module : 2.81 ASE2

Purpose: add socio-economic variables and coefficients in data bank from coding form 2.



input:

parameters from RSE2 and OSE2

IVC = 0 variables
= 1 coefficient

characters forming first field on coding form

characters forming second field on coding form

type = -3 from "all" to "all"
= -1 interzone not "all"
= +1 zone not "all"
= +3 zone "all"
= +4 first field is a key

number of values transmitted

a vector containing either name of keys or name of zones

a vector of new values to add

IACT = 0 if permanent correction
= 1 if temporary modification

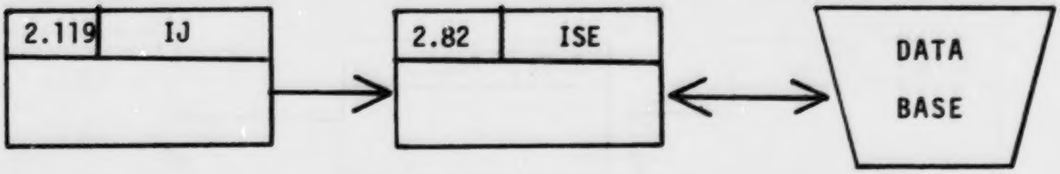
output: see module 2.73 ASE

procedure:

a) if type ≠ 4, reorganize the parameters and call ASE to perform the job.
b) if type = 4, find the key in the file
c) add the new values under the key. Special treatment for the case "all" or "all-all".
d) repeat steps b and c for each key transmitted.

Module : 2.82 ISE

Purpose : expand and renumber socio-economic variables and coefficients



input:

1. file name: SOCIO (file of socio-economic data)
   type: data base
   contents: socio-economic data
2. file name: ZGROUP (file of zone grouping)
   type: data base
   contents: name of groups and definition of groups
3. sorted list of centroid numbers

output:

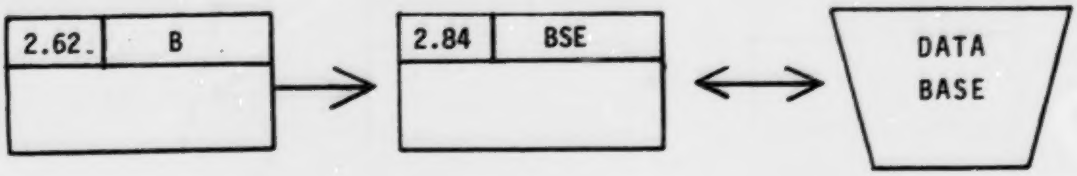
file name: SOCIO (file of socio-economic data)
same file as input

procedure:

a) read in the index file of socio-economic data
b) perform the following steps for each key
c) read in the list of groups and associated values (raw data)
d) with the help of the file of zone grouping expand the groups in their defining zones
e) associate the values to zones, obeying the following precedence:
   (higher) zone itself
   (middle) group
   (lower) "all"
f) write the sorted expanded data in third section of the file and keep track in index. Two formats are defined for expanded data; see file description.

Module : 2.84 BSE

Purpose : clean up file of socio-economic variables and coefficients



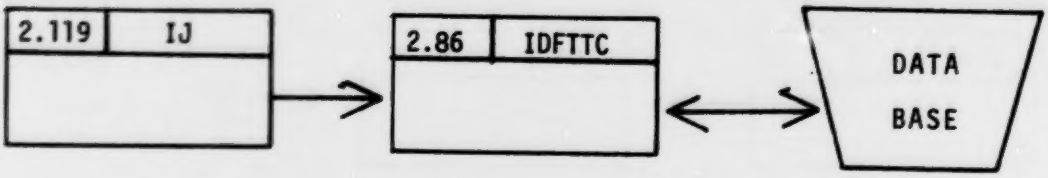
input: file name: SOCIO (file of socio-economic data)
type: data base
contents: socio-economic data

output: same file as input

procedure:
a) read in the index of the file
b) cancel the section of expanded data
c) read the second section (raw data) and eliminate temporary and obsolete data
d) modify the index accordingly.

Module : 2.86 IDFTTC

Purpose : expand and assign internal numbers to demand function travel time coefficients.



input: 1. file name: DCOKIN (file of demand function travel time coeff.)
type: data base
contents: coefficients associated with zones or group of zone
source: user specified

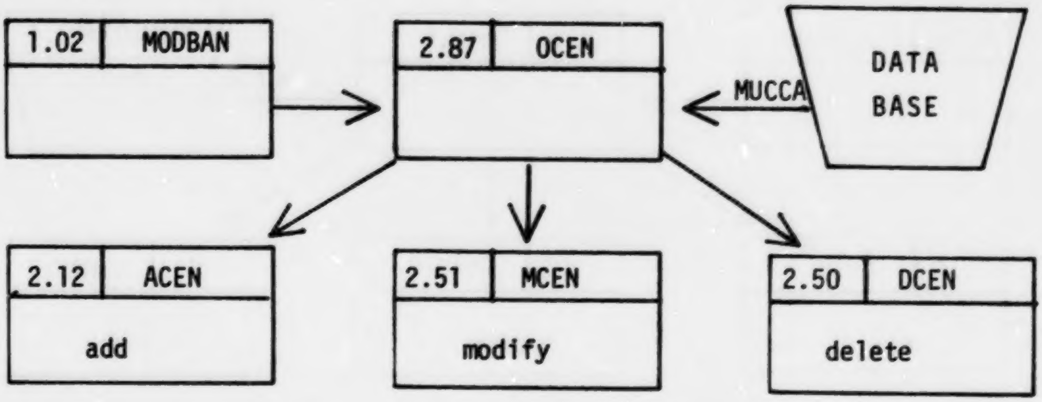
2. sorted list of centroid numbers

output: file name: DCOK (file of demand function travel time coefficients)
type: data base
contents: a vector of coefficients

procedure:
a) clear a vector of length 3 x NCENT
b) read the input file
c) expand the groups
d) set values in the vector, thus performing a direct sort
e) write the vector as file DCOK considering that 3 consecutive words (K0, K1, K2) is a record.

Module : 2.87 OCEN

Purpose : read centroids from a given scenario

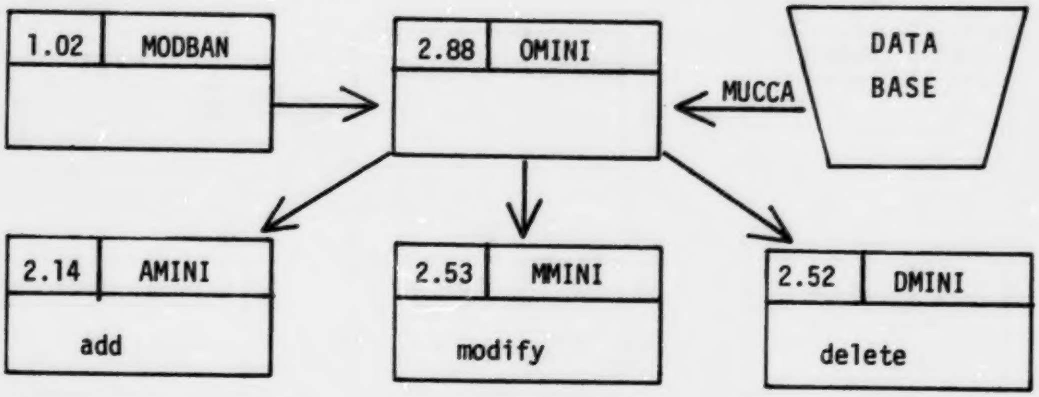

1.02
MODBAN
2.87
OCEN
MUCCA
DATA BASE
2.12
ACEN
add
2.51
MCEN
modify
2.50
DCEN
delete

input: file name: SCEN (file of scenarios)
type: data base
position in index: 61
contents: card images of scenarios

procedure:
a) read in card images from input file (with MUCCA)
b) get all fields without check
c) according to first column, call ACEN, MCEN or DCEN

Module : 2.88 OMINI

Purpose : read transit node - road nodes correspondence from a given scenario

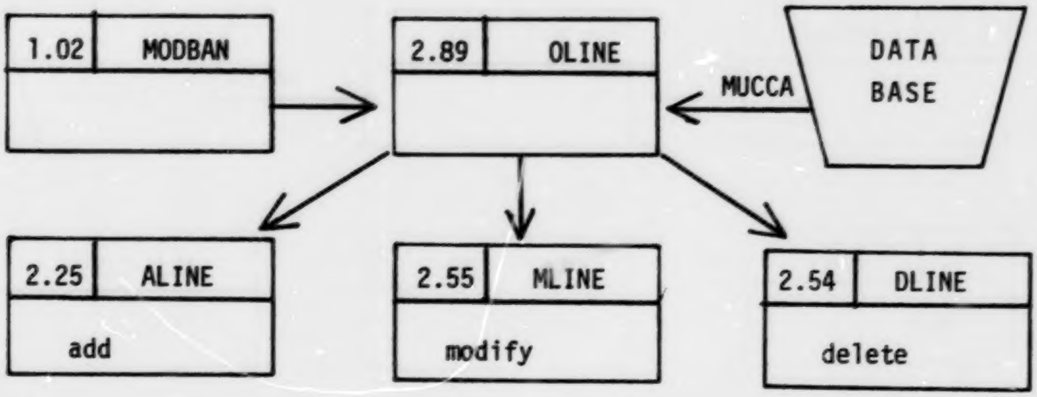

1.02
MODBAN
2.88
OMINI
DATA
BASE
MUCCA
2.14
AMINI
add
2.53
MMINI
modify
2.52
DMINI
delete

input: see module 2.87 OCEN

procedure:
a) call MUCCA to read card images from input file
b) get all nodes without any checking
c) according to first column, call AMINI, MMINI or DMINI

Module : 2.89 OLINE

Purpose : read transit lines from a given scenario

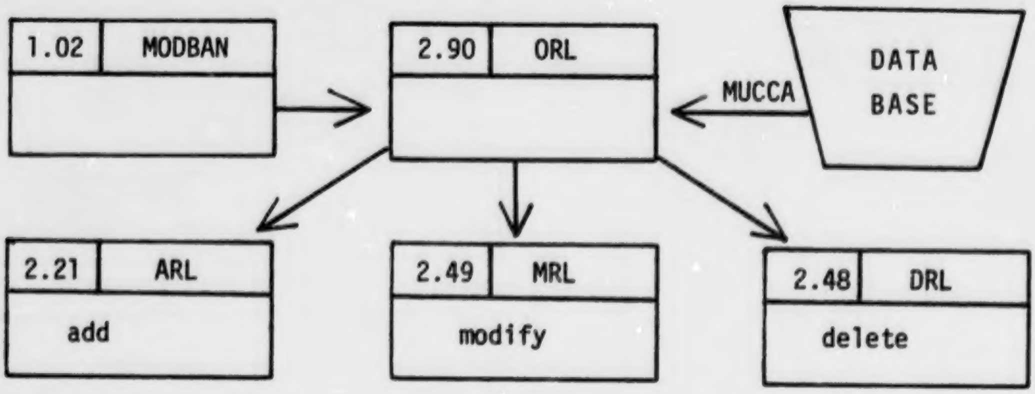

1.02
MODBAN
2.89
OLINE
MUCCA
DATA BASE
2.25
ALINE
add
2.55
MLINE
modify
2.54
DLINE
delete

input: see module 2.87 ICEN

procedure:
a) call MUCCA to read card images from input file
b) get characteristics of the transit line
c) read following cards to get the nodes along the route
d) it is not necessary to check anything
e) according to first column of first card, call ALINE, MLINE or DLINE.

Module : 2.90 ORL

Purpose : read road links from a given scenario

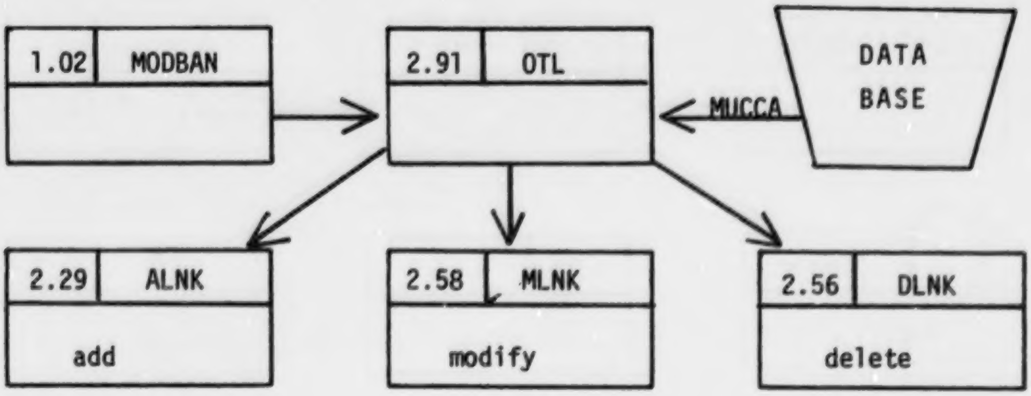

1.02
MODBAN
2.90
ORL
MUCCA
DATA BASE
2.21
ARL
add
2.49
MRL
modify
2.48
DRL
delete

input: see module 2.87 OCEN

procedure:
a) call MUCCA to read card images from input file
b) get all fields without checking them: origin, destination, type, length, observed time, observed volume and data
c) call ARL, MRL or DRL
d) in case of symmetry, repeat c).

Module : 2.91 OTL

Procedure : read transit links from a given scenario

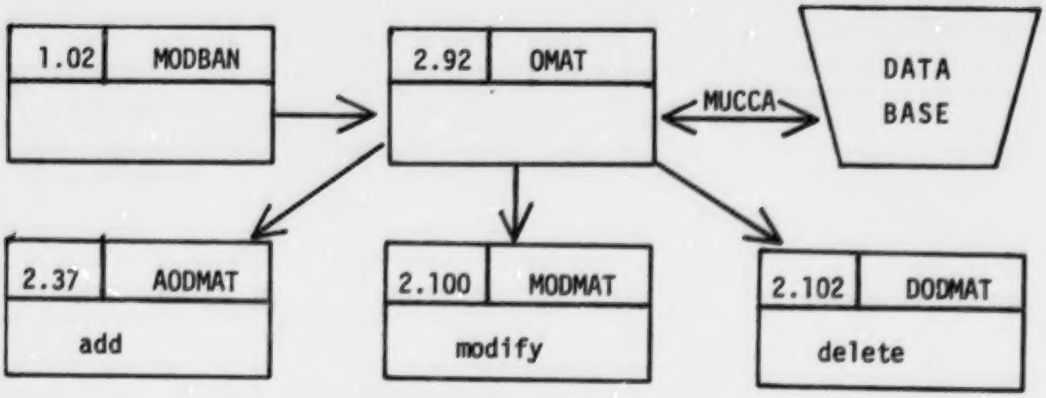

1.02
MODBAN
2.91
OTL
DATA BASE
MUCCA
2.29
ALNK
add
2.58
MLNK
modify
2.56
DLNK
delete

input:
1. see module 2.87 OCEN
2. parameter from MODBAN indicating "transit links" or "transit access links"

procedure:
a) call MUCCA to read card images from input file
b) get origin, destination and times without checking them
c) call ALNK, MLNK or DLNK
d) in case of symmetry, repeat step c).

Module : 2.92 OMAT

Purpose : read a matrix from a given scenario

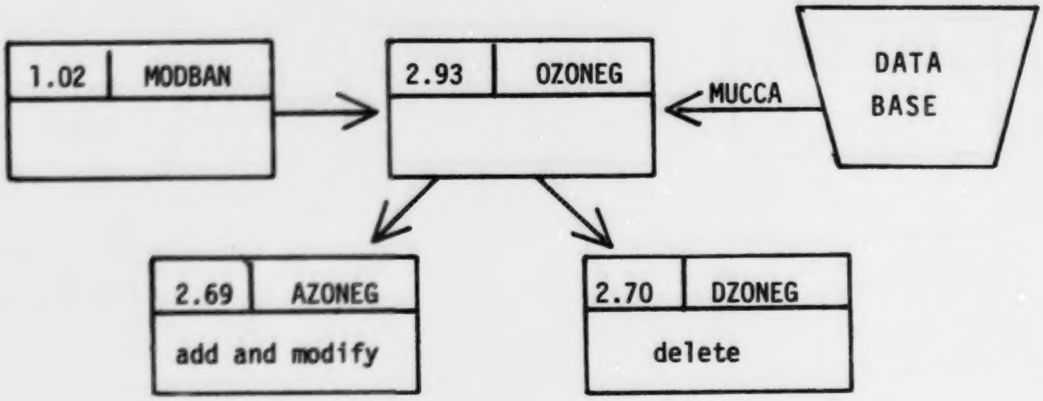

1.02
MODBAN
2.92
OMAT
MUCCA
DATA
BASE
2.37
AODMAT
add
2.100
MODMAT
modify
2.102
DODMAT
delete

input:
1. see module 2.87 OCEN
2. parameter from MODBAN indicating which of the 6 matrices.

procedure:
a) call MUCCA to read card images from input file
b) get all fields and construct vectors of destination and values
c) call AODMAT, MODMAT or DODMAT.

Module : 2.93 OZONEG

Purpose : read zone grouping from a given scenario

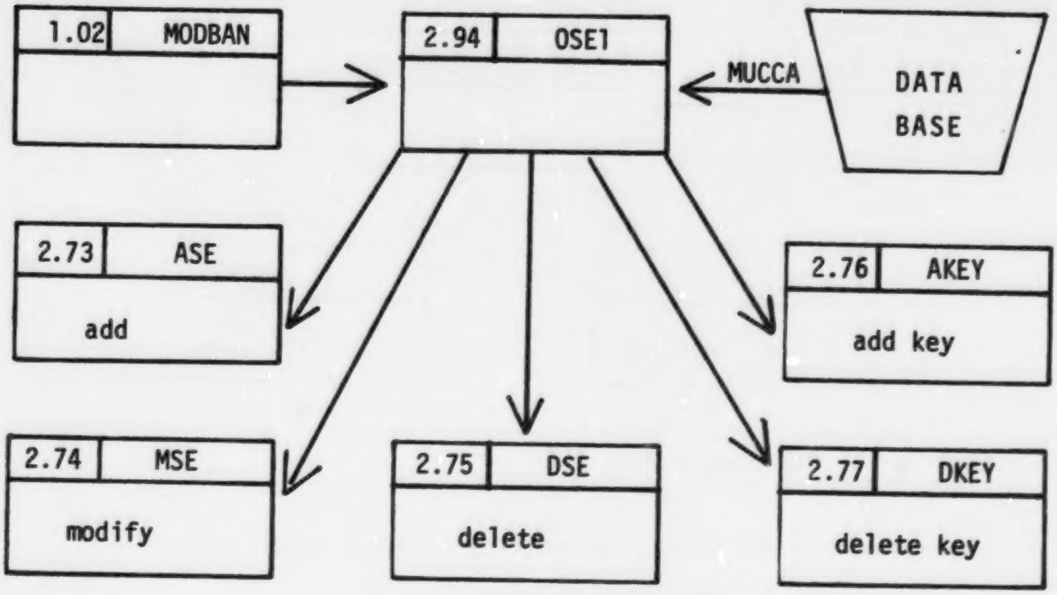

1.02
MODBAN
2.93
OZONEG
MUCCA
DATA BASE
2.69
AZONEG
add and modify
2.70
DZONEG
delete

input: see module 2.87 OCEN

procedure:
a) call MUCCA to read card images from input file
b) accumulate zones forming a group
c) call AZONEG or DZONEG.

Module : 2.94 OSE1

Purpose : read socio-economic variables and coefficients from a given scenario, on coding form 1.


1.02
MODBAN
2.94
OSE1
MUCCA
DATA BASE
2.73
ASE
add
2.76
AKEY
add key
2.74
MSE
modify
2.75
DSE
delete
2.77
DKEY
delete key

input:

1. parameter from MODBAN indicating variables or coefficients
2. see module 2.87 OCEN

procedure

a) call MUCCA to read card images from input file

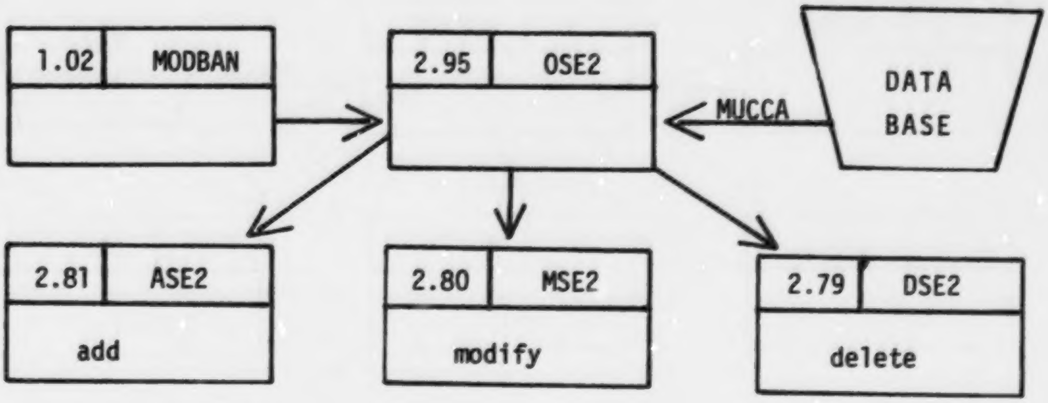
b) if it is a "keys" card, get the zone or interzone keys and call AKEY or DKEY

c) if it is not a "keys" card, determine the type of variables or coefficients and accumulate the values

d) call ASE, MSE or DSE.

Module : 2.95 OSE2

Purpose : read socio-economic variables and coefficients from a given scenario, on coding form 2.

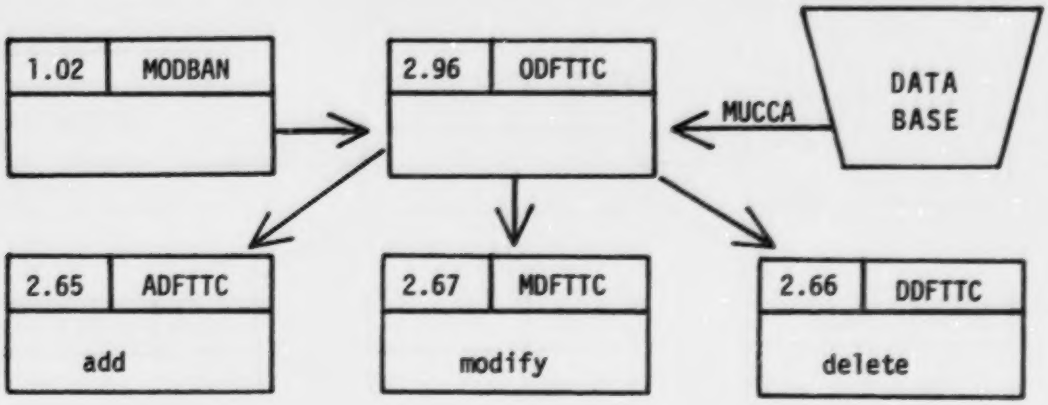

1.02
MODBAN
2.95
OSE2
MUCCA
DATA BASE
2.81
ASE2
add
2.80
MSE2
modify
2.79
DSE2
delete

input:
1. parameter from MODBAN indicating variables or coefficients.
2. see module 2.87 OCEN.

procedure:
a) call MUCCA to read card images from input file
b) determine if the data are entered by zone or by key name
c) when they are entered by zone, accumulate key names and values, check also the presence of a destination zone
d) when they are entered by key name, find the origin zone and its value or accumulate the list of pairs O-D with their values
e) call ASE2, MSE2 or DSE2.

Mpdule : 2.96 ODFTTC

Purpose : read demand function travel time coefficients from a given scenario

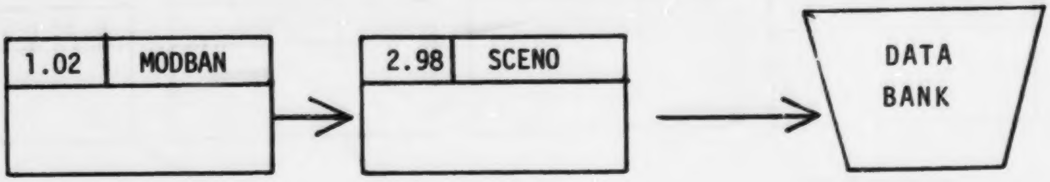

1.02
MODBAN
2.96
ODFTTC
MUCCA
DATA
BASE
2.65
ADFTTC
add
2.67
MDFTTC
modify
2.66
DDFTTC
delete

input: see module 2.87 OCEN

procedure: a) call MUCCA to read card images from input file
b) get zone numbers and 3 coefficients
c) call ADFTTC. MDFTTC or DDFTTC.

Module: 2.98 SCENO

Purpose: define a scenario and prepare accumulation



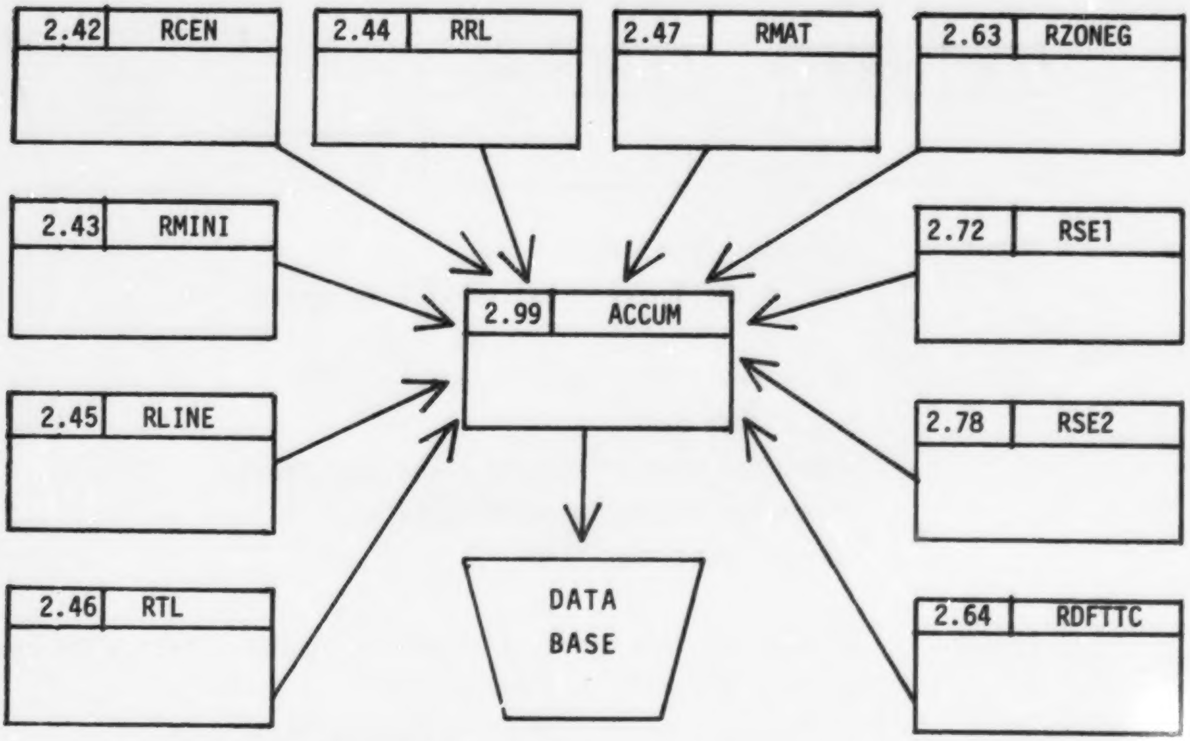
input: parameter from MODBAN
name of the scenario as supplied by user

output: file name: SCEN (file of scenarios)
type: data bank
contents: name of scenarios and card images
sequence: sequential with index
source: used specified

procedure:
a) check if this scenario is already defined: search the index
b) write the name in index
c) open up a block of data to receive card images
d) place in index a pointer to the open block.

Module : 2.99 ACCUM

Procedure : accumulate card images of a scenario


2.42 RCEN
2.44 RRL
2.47 RMAT
2.63 RZONEG
2.43 RMINI
2.72 RSE1
2.99 ACCUM
2.45 RLINE
2.78 RSE2
2.46 RTL
DATA BASE
2.64 RDFTTC

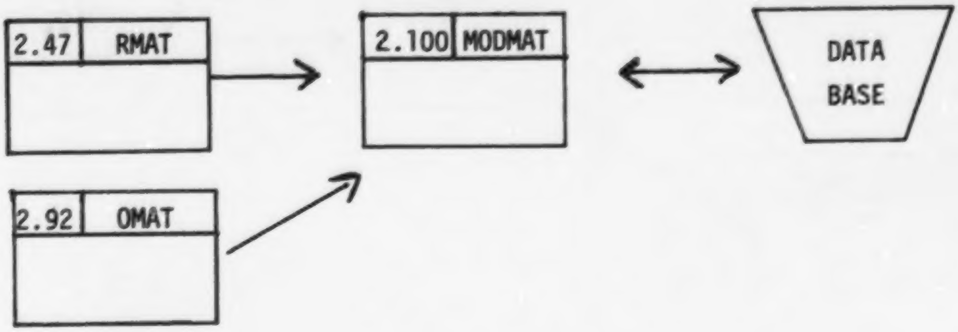
input: a COMMON containing card image belonging to a scenario

output: file name: SCEN (file of scenarios)
type: data base
contents: card images read from COMFIL
sequence: sequential with index
source: user specified

procedure:
a) count the characters on the line
b) if line is empty, exit
c) count the words needed to contain the characters of the line
d) pack these characters as a record of the file.

Module : 2.100 MØDMAT

Purpose : Modify a matrix from information supplied by user



Input : parameters from RMAT and OMAT:

origin node (row of the matrix)
number of destinations
vector of destination nodes
vector of values associated with destinations
flag indicating permanent correction / temporary modification
identification of the matrix

Output : see 2.37 AODMAT

Procedure : a) find the row (origin nodes)

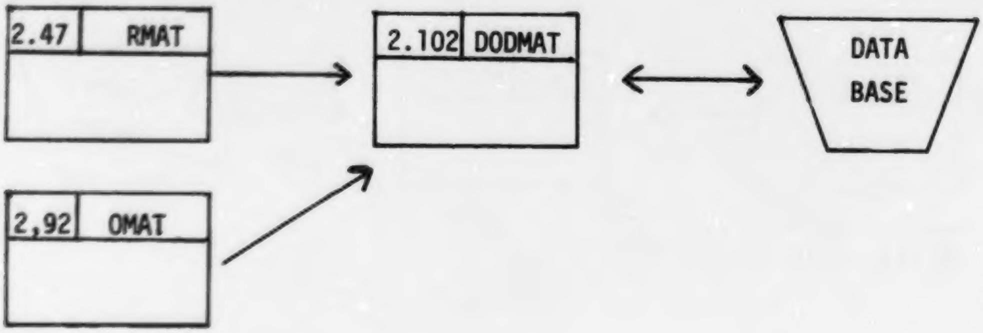
b) if the row is not defined, add new values

c) when the row is defined, find columns

d) replace values where they are if possible; add them at end of file if not.

Module : 2.102 DODMAT

Purpose : Delete values in an O-D matrix



Input : Parameters from RMAT and OMAT:

    origin or row of the matrix
    number of destinations to be deleted, possibly all
    vector of destinations
    flag indicating permanent correction / temporary modification
    identification of the matrix

Output : See module 2.37 AODMAT

Procedure : a) find the row (origin node)

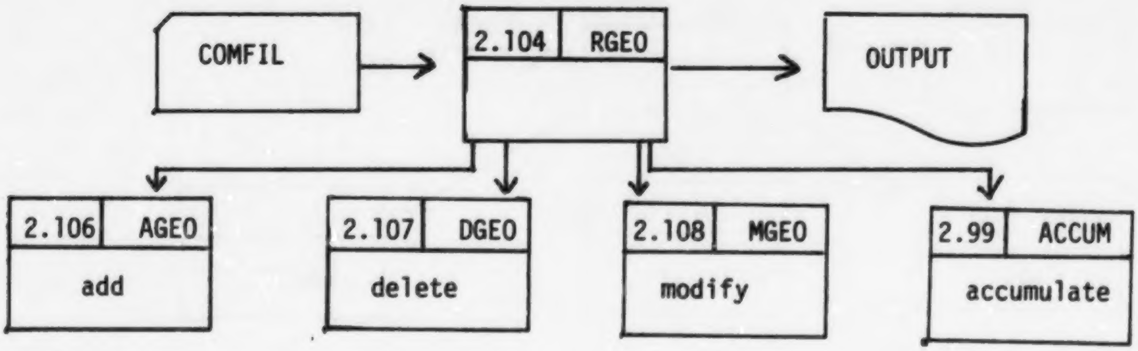
b) cancel the whole row if number of destinations is 0

c) find the columns (destinations nodes) when their number is greater than 0

d) cancel values.

Module : 2.104 RGEO

Purpose : Read node coordinates from information supplied by user


COMFIL
2.104
RGEO
OUTPUT
2.106
AGEO
add
2.107
DGEO
delete
2.108
MGEO
modify
2.99
ACCUM
accumulate

Input : File name: COMFIL

Type : card images

Contents : see "node coordinate coding form"

Sequence : sequential

Source : user specified

Output : File name: OUTPUT

Type : report file

Status : will be routed to printer

Contents : listing of input cards and error messages.

Procedure : a) read in card images

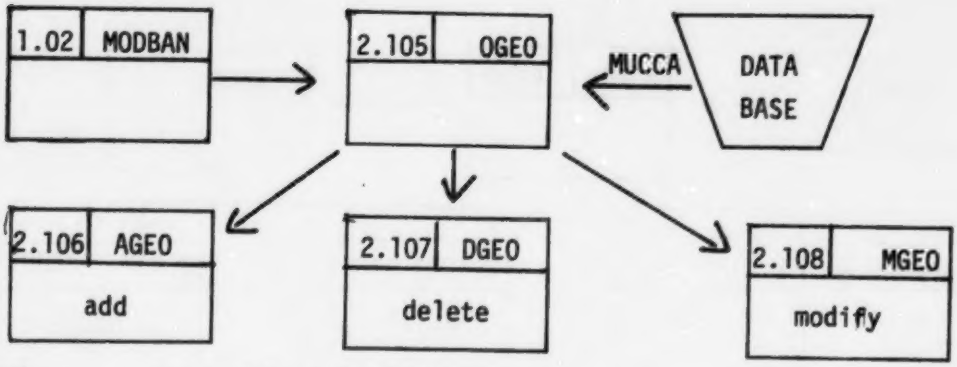
b) check validity of all fields

c) in correction mode, call AGEO, MGEO or DGEO

d) in scenario mode, call ACCUM

Module : 2.105 OGEO

Purpose : Read node coordinates from a given scenario.


1.02
MODBAN
2.105
OGEO
MUCCA
DATA BASE
2.106
AGEO
add
2.107
DGEO
delete
2.108
MGEO
modify

Input : see module 2.87 OCEN

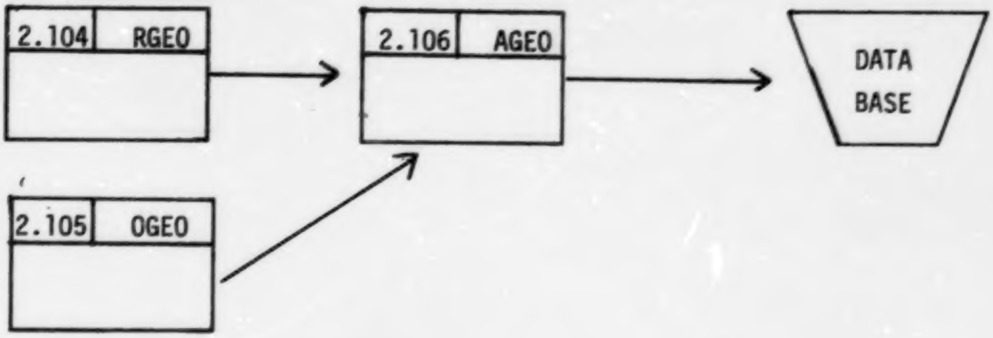
Procedure : a) read in card images from data base

b) get node numbers and coordinates

c) according to first column, call AGEO, MGEO or DGEO.

Module : 2.106 AGEO

Purpose : Add node coordinates to data bank.



Input : Parameters from RGEO and OGEO:

node number
X-coordinate
Y-coordinate
flag of permanent correction / temporary modification

Output : File name: NODEIN (file of geocoded nodes)

Type : data base

Contents : node numbers and their coordinates

Sequence : sequential

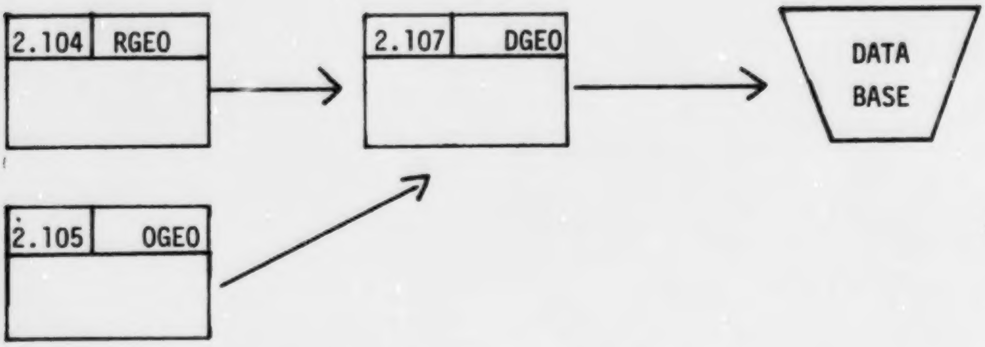
Position in index: 4

Procedure : a) check if already defined

b) if not add at end of file.

Module : 2.107 DGEO

Purpose : Delete node coordinates from data bank


2.104 RGEO
2.107 DGEO
DATA BASE
2.105 OGEO

Input : Parameters from RGEO and OGEO:
node number
flag of permanent correction / temporary modification

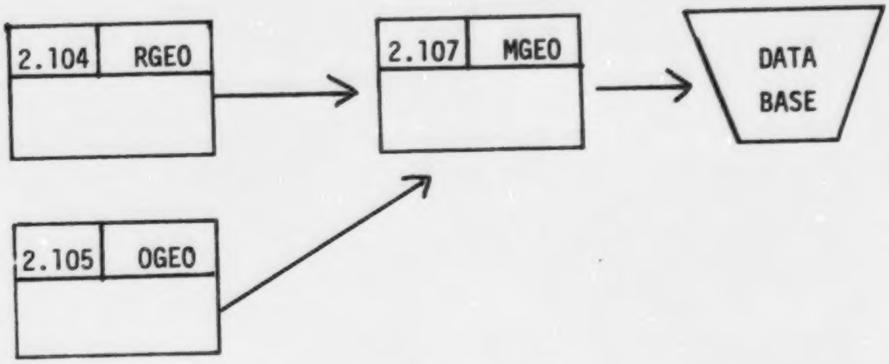
Output : See module 2.106 AGEO

Procedure : a) check if this node is defined

b) mark it as inactive.

Module : 2.108 MGEO

Purpose : Modify node coordinates in data bank



Input : Parameters from RGEO and OGEO:

node number
new X-coordinate
new Y-coordinate
indicators which tell if X- and Y- coordinates have to be changed
flag of permanent correction / temporary modification

Output : See module 2.106 AGEO

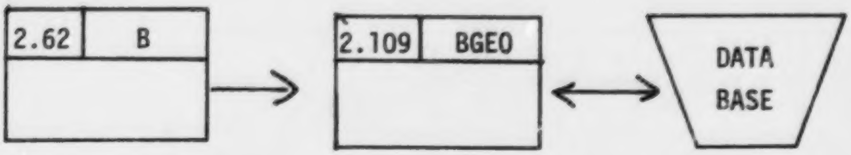
Procedure : a) check if this node is defined

b) change coordinates in same record if possible

c) if not, mark them as inactive and add new ones at end of file.

Module : 2.109 BGEO

Purpose : Clean up file of coordinates



Input : 1. File name: NODEIN (file of node coordinates)

Type : data base

Position in index: 4

Source : created by 2.106 AGEO
2.107 DGEO
2.108 MGEO

2. File name: NODES (file of node coordinate)

Type : data base

Position in index: 5

Source : created by 2.110 IGEO

Output : The first input file.

Procedure : a) eliminate the second input file

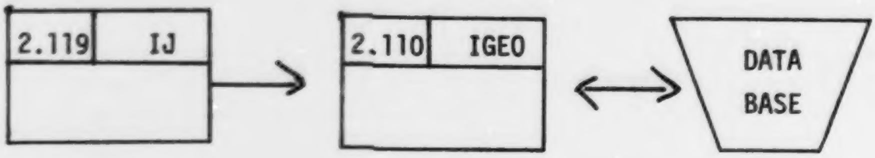
b) read the first one, record by record

c) eliminate temporary and obsolete data

d) rewrite clean records.

Module : 2.110 IGEO

Purpose : Expand node coordinates, renumber and sort.



Input : File name: NODEIN (file of geocoded nodes)

Type : data base

Contents : node numbers and coordinates

Position in index: 4

Source : created by 2.106 AGEO
2.107 DGEO
2.108 MGEO

Output : File name: NODES (file of geocoded nodes)

Type : data base

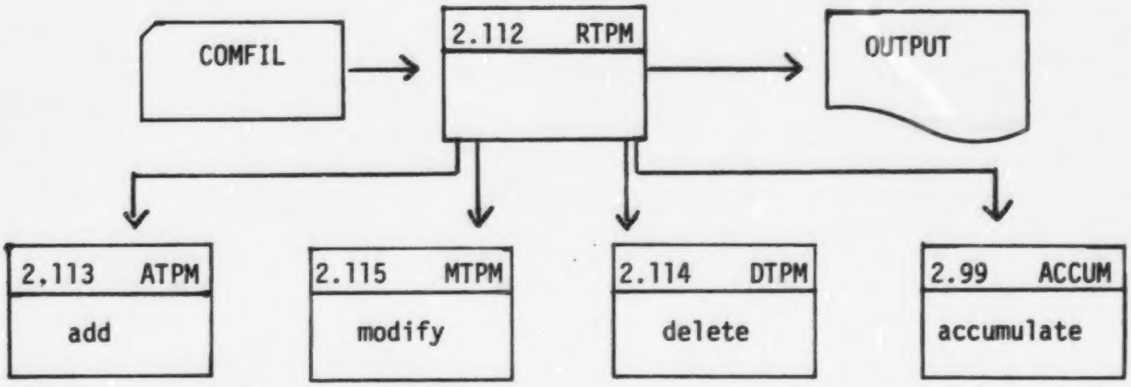
Contents : a vector of coordinates

Position in index: 5

Procedure :
a) read input file
b) assign internal numbers to node numbers
c) sort vectors of internal numbers with their coordinates
d) create the output file; copy on this file only coordinates. The position of coordinate is the internal node number.

Module : 2.112 RTPM

Purpose : Read transit path multipliers from information supplied by user.


COMFIL
2.112 RTPM
OUTPUT
2.113 ATPM
add
2.115 MTPM
modify
2.114 DTPM
delete
2.99 ACCUM
accumulate

Input : File name: COMFIL

Type : direct entry input file

Contents : see "transit path multipliers coding form"

Sequence : sequential

Source : user specified.

Output : File name: OUTPUT

Type : report file

Status : will be routed to printer

Contents : listing of input cards and error messages.

Procedure : a) read in card images

b) get zone number and accumulate 2 coefficients

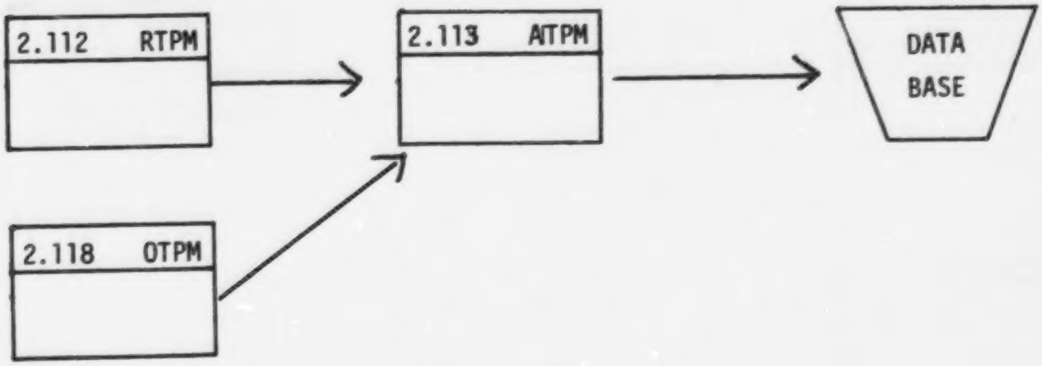
c) check validity of all fields

d) in modification mode, call ATPM, MTPM or DTPM

e) in scenario mode, call ACCUM

Module : 2.113 ATPM

Purpose : Add transit path multipliers to data bank



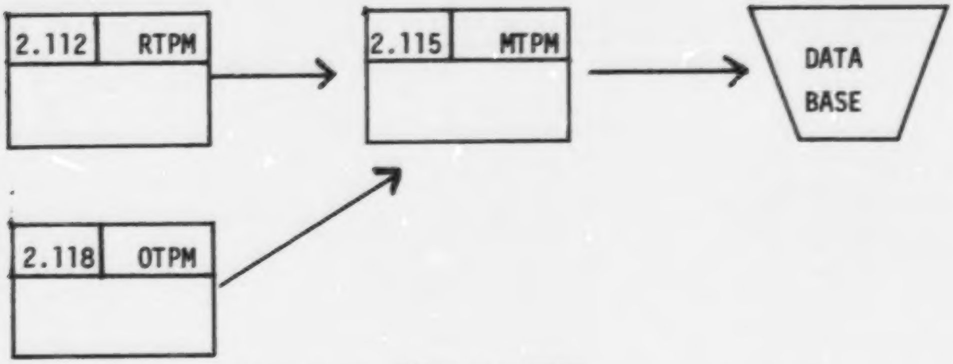
Input : Parameters from RTPM and OTPM:
name of zone
2 coefficients
flag permanent correction / temporary modification

Output : File name: PMUWIN (file of transit path multipliers)
Type : data base
Contents : coefficients associated with zones or groups of zones
Sequence : sequential
Use : input to 2.117 ITPM
2.116 BTPM

Procedure : a) check if already defined
b) add new coefficients

Module : 2.115 MTPM

Prupose : Modify transit path multipliers from information supplied by user.



Input : Parameters from TRPM and OTPM:

name of zone

2 coefficients

2 indicators set to true if corresponding coefficients are to be changed

activity (permanent / temporary)

Output : See module 2.113 ATPM

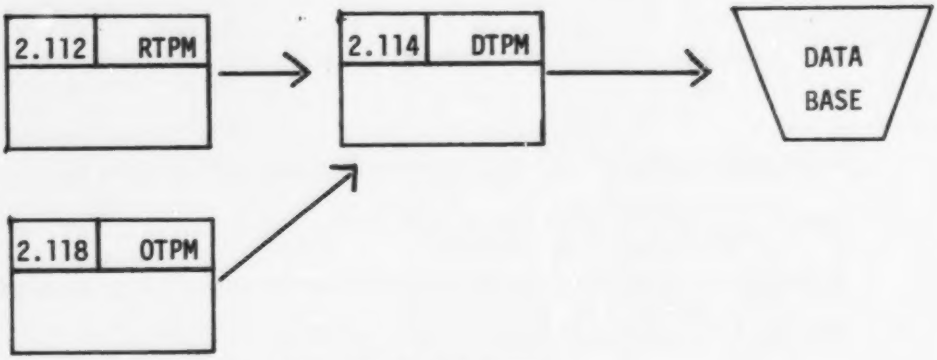
Procedure : a) find the zone in the file

b) if possible, change the coefficients in the record

c) if not, mark the record as inactive and add a new one at end of file.

Module : 2.114 DTPM

Purpose : Delete transit path multipliers from data bank.


2.112
RTPM
2.114
DTPM
DATA
BASE
2.118
OTPM

Input : Parameters from RTPM and OTPM:
name of group
active indicator (permanent/temporary)

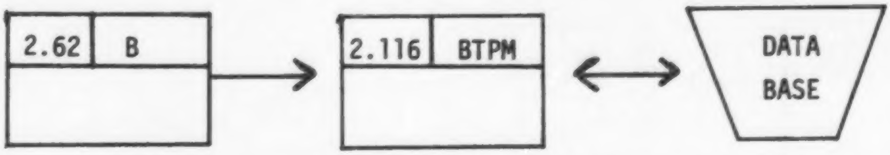
Output : See module 2.113 ATPM

Procedure : a) find the group or zone in the file

b) mark it as inactive.

Module : 2.116 BTPM

Purpose : Clear up file of transit path multipliers.



Input : 1. File name: PMUWIN (file of transit path multipliers)

Type : data base

Contents : coefficients associated with zones or groups of zones

Source : created by: 2.113 ATPM
2.114 DTPM
2.115 MTPM

2. File name: PMUW (file of expanded transit path multipliers)

Type : data base

Contents : a vector of coefficients

Source : constructed by ITPM

Output : Same as the first input file.

Procedure : a) eliminate the second input file

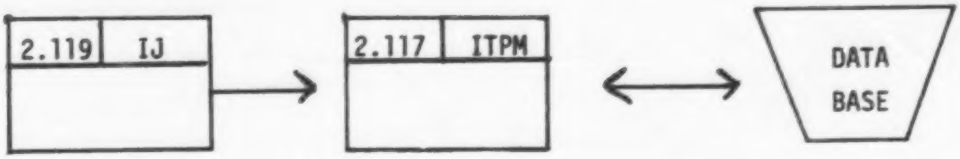
b) read the first one, record by record

c) eliminate temporary and obsolete data

d) rewrite clean records

Module : 2.117 ITPM

Purpose : Expand and assign internal numbers to transit path multipliers.



Input : 1. File name: PMUWIN (file of transit path multipliers)

Type : data base

Contents : coefficients associated with zones or groups of zones

Source : created by 2.113 ATPM
2.115 MTPM
2.114 DTPM

2. Sorted list of centroid numbers

3. File name: ZGROUP (file of zone grouping)

Type : data base

Contents : definition of groups of zones

Source : user specified

Output : File name: PMUW (file of expanded transit path multipliers)

Type : data base

Contents : a vector of coefficients

Procedure : a) clear a vector of length 2 x NCENT

b) read the input files

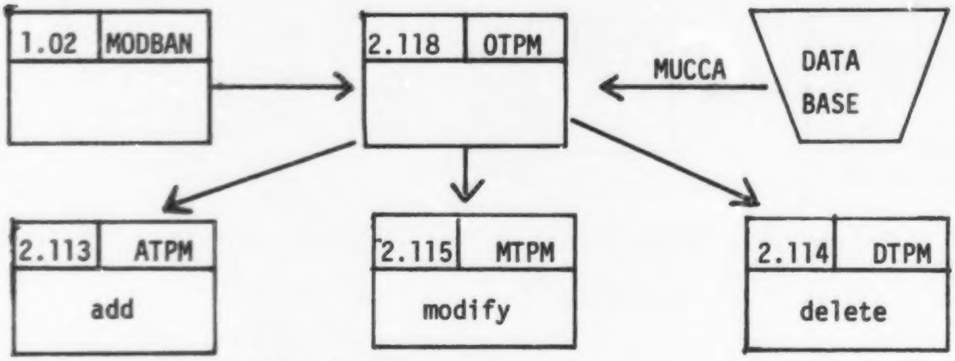
c) expand the groups

d) set values in the vector

e) write the vector on output file, considering that 2 consecutive words constitute a record.

Module : 2.118 OTPM

Purpose : Read transit path multipliers from a given scenario.


1.02
MODBAN
2.118
OTPM
MUCCA
DATA BASE
2.113
ATPM
add
2.115
MTPM
modify
2.114
DTPM
delete

Input : See module 2.87 OCEN

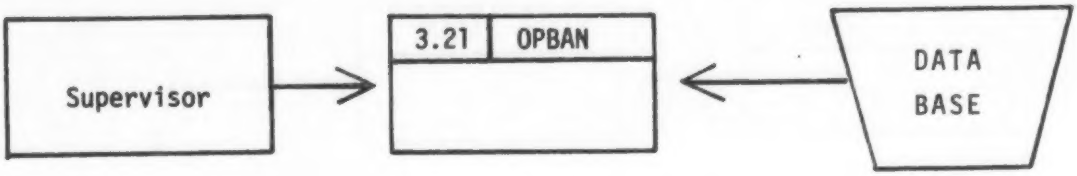
Procedure : a) call MUCCA to read card images from input file

b) get zone number and 2 coefficients

c) call ATPM, MTPM or DTPM according to request.

Module : 3.21 OPBAN

Purpose : open data bank if not yet opened and initialize



input: 1. file name: NODEXT (file of internal-external numbers)
type: data base
position in index: 54

2. data base

output: 1. vector of centroid numbers

2. vector of turning nodes

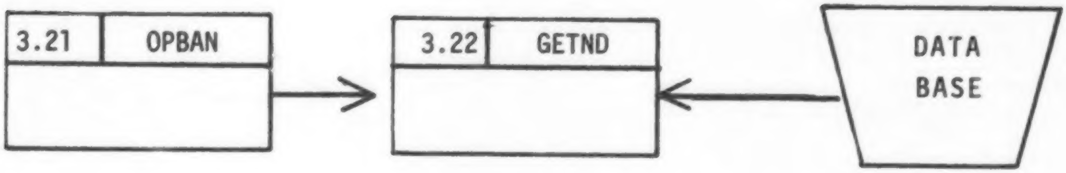
3. vector of regular road nodes

4. vector of transit-only nodes

procedure:
a) if data base is not opened, yet, open the file and read in the first record, which is the index
b) if position 53 of the index is non-zero, the data bank is not in working condition. Stop
c) if data base was not opened at the beginning of the module, call GETND 4 times to obtain the 4 vectors
d) initialize the common area / NETSIZ /

Module : 3.22 GETND

Purpose : read a vector of external node numbers from databank



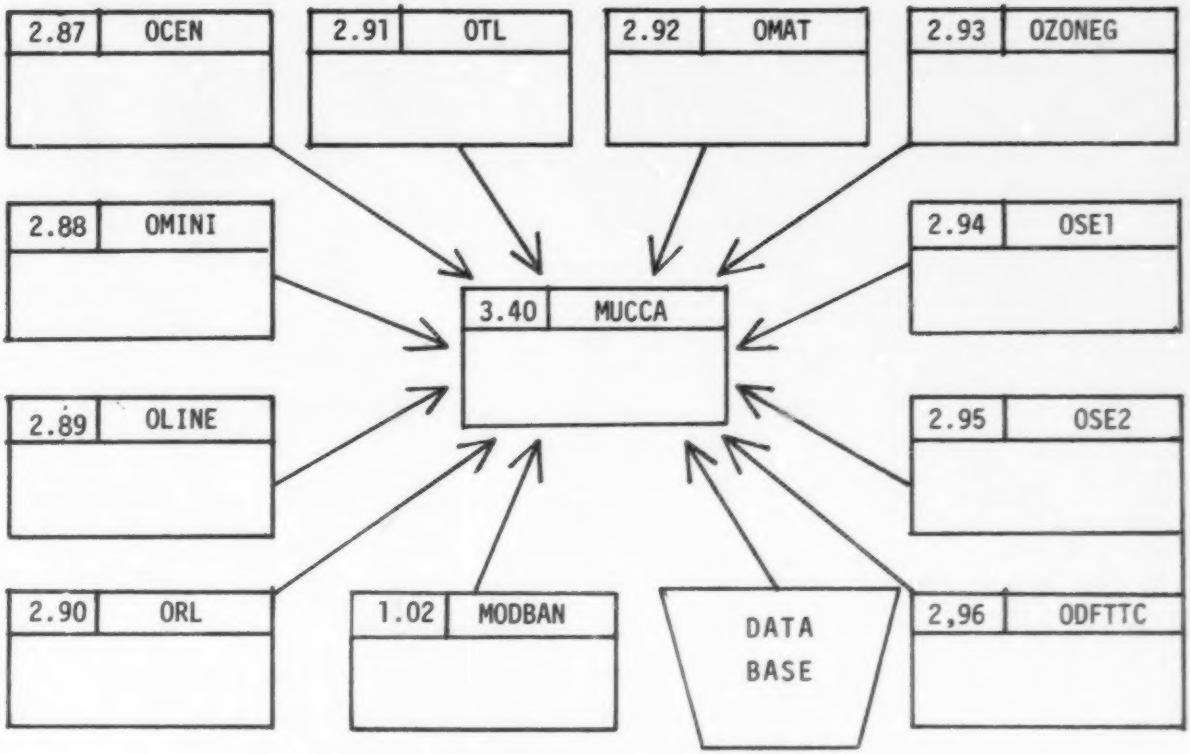
input: file name: NODEXT (file of internal-external numbers)
type: data base
position in index: 54

output: a vector and its length

procedure:
a) read in next record
b) transfer first word as length of the vector
c) if it is 0, exit
d) reserve an area in core for the vector
e) transfer to vector following words of record

Module : 3.40 MUCCA

Purpose : deliver next card image from a kept scenario


2.87 OCEN
2.91 OTL
2.92 OMAT
2.93 OZONEG
2.88 OMINI
2.94 OSE1
3.40 MUCCA
2.89 OLINE
2.95 OSE2
2.90 ORL
1.02 MODBAN
DATA BASE
2,96 ODFTTC

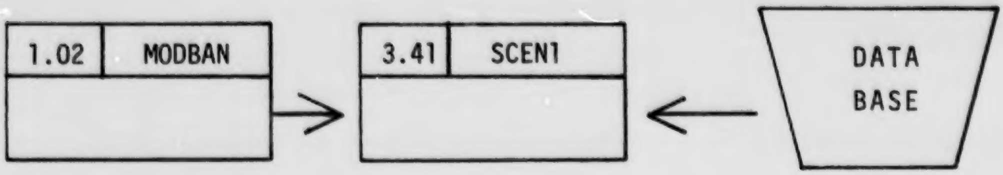
input: file name: SCEN (file of scenarios)
type: data bank
contents: packed card images of a scenario
sequence: sequential with index

output: a COMMON containing card image as if read from file COMFIL

procedure: a) erase the COMMON area
b) get the count of characters of the next card image
c) unpack the next card image into the COMMCN area.

Module : 3.41 SCEN1

Purpose : check if a scenario is defined and put it in the stack



input:

1. parameter from MODBAN
   name of a scenario
2. file name: SCEN (file of scenarios)
   type: data base
   contents: name of scenarios and card images

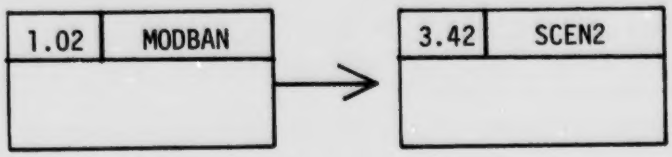
output: a stack of pointers to scenarios to be applied

procedure:

a) read the index of the file
b) check if scenario is defined
c) if the purpose was checking the existence only, exit
d) stop the application of the current scenario, if any. Place in the stack the current position of the file
e) place in the stack the position of the found scenario and mark it as not yet applied.

Module : 3.42 SCEN2

Purpose : initialize next scenario



input: the stack of pointers to scenarios to be applied

output: same as input

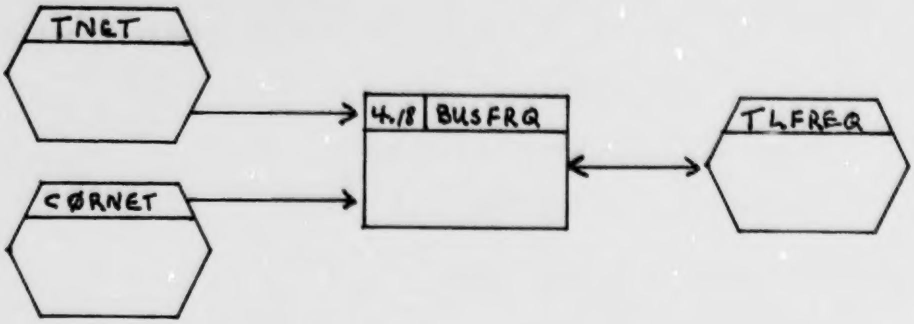
procedure:
a) check if there is a newly added scenario on top of the stack
b) if yes, read in first block and set pointer to first record
c) otherwise, push down the stack
d) if stack is empty, exit
e) if not empty, read in the block containing next record of the stacked scenario.

note: the idea is to allow inclusion of a scenario within another.

MODULE	: 4.00 ASSPRO
PURPOSE	: Control assignment procedure according to assignment options.
INPUT	: Data in COMMON area /APARM/ Assignment parameters set as described in modules 1.60 DECASS and 1.62 DECPAR
OUTPUT	: Subroutine parameter LOGICAL OK TRUE if all modules called by ASSPRO executed correctly. FALSE if a module encountered an error and returned OK =FALSE.
PROCEDURE	: See comments in the module code and flowchart.

MODULE : 4.18 BUSFRQ

PURPOSE : Determine bus frequency by road link



INPUT : 1. File name : TNET (515)

Type : Intermediate file

Symbol :

Status : Required

Contents : Vectors representing the transit network structure:

FLIN, NOLIST, NONLIN, LISTN, NDIR, DA, DR, DISA, DISR, LISTL, LPNT, NADR

Sequence : Sequential

Source : Output from 5.15 TTIME and updated in 4.61 CAPCHK

2. File name : CØRNET (514)

Type : Intermediate file

Symbol :

Status : Required if TLFREQ file is created for the first time (i.e. not required for updating)

Contents : Vectors representing the transit line itineraries as sequences of road links: NT, NA, NR, NOT(I), NOA(J), LINKA(J), NOR(K), LINKR(K)

Sequence : Sequential by line

Source : Output from 5.14 EXPNET

3. File name : TLFREQ (518)

Type : Intermediate file

Symbol : f_a^{tr}

Status : Required if the update option is used

Contents : Number of vehicles per hour on each road link used by transit vehicles and a list of all transit lines using that link

Sequence : Sequential by link

Use : To construct directly the list of transit lines on road links

OUTPUT : File name : TLFREQ (see above)

Status : Always created

Use : To define transit vehicle flow on road links in 6.50 BUSIN

PROCEDURE : a) Construct list LINEON which will contain for each road link a list of transit lines using it:

Case 1: creation of file TLFREQ

- for each line read in vectors NOA, LINKA, NOR, LINKR from CORNET
- determine road links used by the line from LINKA and LINKR save their origin and destination nodes in vectors NO(L) and ND(L) and insert the transit line number in the list

Case 2: update file TLFREQ

- read in the list of lines on road links from file TLFREQ

b) Read in the vector of transit headways FLIN from file TNET and convert headways to frequencies

c) Using list LINEON, accumulate transit frequencies in vector FATR(L)

d) For each road link L with a non-zero bus frequency, write on TLFREQ a record containing NO(L), ND(L), FATR(L) and the list of transit lines on that link

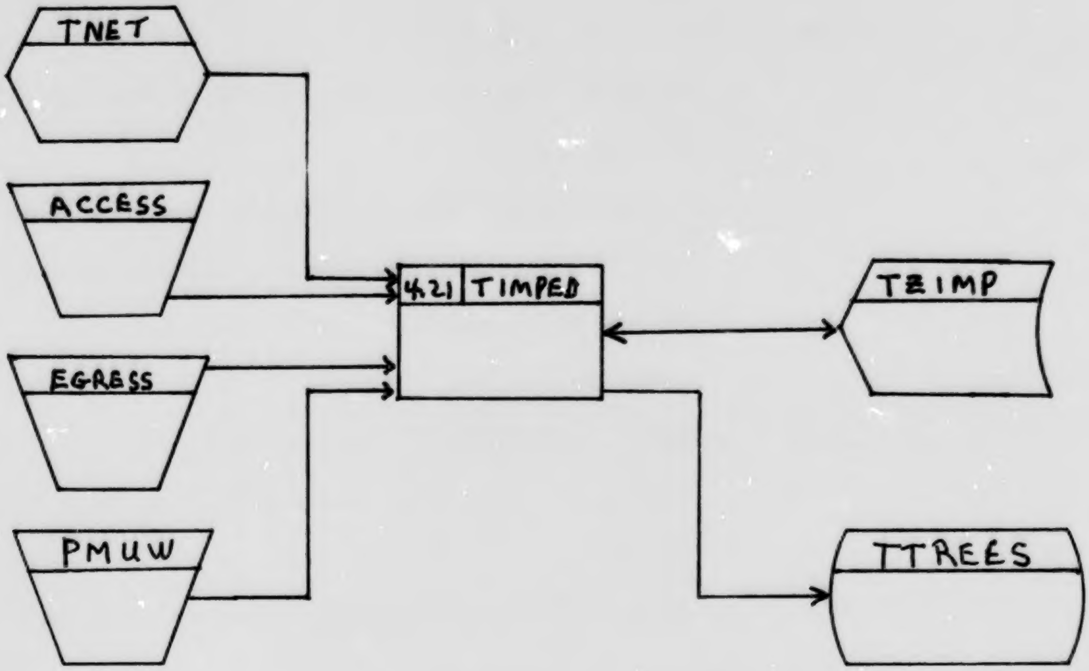
NOTE: Subroutine BUSFRQ contains two entry points:

BUSFRQ: create new TLFREQ file

UPTLFR: update existing TLFREQ file

MODULE : 4.21 TIMPED

PURPOSE : Calculate new transit impedance matrix and compare


TNET
ACCESS
EGRESS
PMUW
4.21
TIMPED
TZIMP
TTREES

INPUT : 1. File name : TNET (515)

Type : Intermediate file

Symbol :

Status : Required

Contents : Vectors representing the transit network structure:

FLIN, NOLIST, NONLIN, LISTN, NDIR, DA, DR, DISA, DISR, LISTL, LPNT, NADR

Sequence : Sequential

Source : Output from 5.15 TTIME and updated in 4.31 TIMCHK

2. File name : ACCESS (203)

Type : Data base

Symbol :

Status : Required

Contents : Access links from zone centroids to transit network nodes and corresponding access time

Sequence : Sequential, sorted by the data base service modules in ascending order by zone centroid

Source : User specified

3. File name : EGRESS (203)

Type : Data base

Symbol :

Status : Required

Contents : Egress links from transit network nodes to zone centroids and corresponding egress time

Sequence : Sequential, sorted by the data base service modules in ascending order by zone centroid

Source : User specified

4. File name : PMUW (250)

Type : Data base

Symbol : $w1_p$, $w2_p$

Status : Optional

Contents : Transit path coefficients multipliers for each origin p:

-$w1_p$ is the multiplier for walk time coefficient

.walking impedance = $w1_p$* WALK * access time

-$w2_p$ is the multiplier for waiting time coefficient

($w2_p$ * WAIT)

Sequence : Sequential by origin p , p = 1, NCENT

Source : User specified

5. File name : TZIMP (531)

Type : Intermediate file

Symbol : u^{tr}_{pq}

Status : Required if new transit impedances are to be compared with previous ones

Contents : Vector of transit impedance from origin p to all destinations q , q =1, NCENT

Sequence : Random by origin p , p = 1, NCENT

Use : To input previous impedances.

OUTPUT : 1. File name : TZIMP (see above)

Status : Created or updated

Use : To calculate demand function component K^{au}_{pq} in 6.25 CALCK

2. File name : TTREES (521)

Type : Work file

Symbol :

Status : Always created

Contents : Shortest path tree from each origin p to all transit network nodes n , n=1, NNODET

Sequence : Sequential by origin p , P=1, NCENT. If an origin p is not linked to the transit network, the record corresponding to that origin is not written onto file.

Use : In the case of bimodal assignment only module 5.50 TASS will read shortest paths from file TTREES instead of recomputing them.

PROCEDURE : 1) Read in the transit network from file TNET:

NLINE, NSEG, NSEL and vectors

FLIN, NOLIST, NONLIN, LISTN, NDIR, DA, DR, DISA, DISR, LISTL, LPNT, NADR

2) Read in access links from file ACCESS and construct the following vectors:

NAL = number of access links

NAC(p) = number of first access link from centroid p ,
p = 1, NCENT

NAC(NCENT + 1) = NAL + 1

NODAC(L) = destination node of access link L, L = 1, NAL

ACT(L) = access time on access link L, L = 1, NAL

3) Read in egress links from file EGRESS and construct the following vectors:

NEL = number of egress links

NEG(p) = number of first egress link to centroid p,
p = 1, NCENT

NEG(NCENT + 1) = NEL + 1

NODEG(L) = origin node of egress link L,L = 1, NEL

EGT(L) = egress time on egress link L, L = 1, NEL

4) Save path coefficients
ZWALK = WWALK
ZWAIT = WWAIT
and rewind path file TTREES

5) For each origin p,

a) If path multipliers w are to be used, read $w1_p$, $w2_p$ from file PMUW and define path coefficients for origin p

$WWALK = ZWALK * w1_p$

$WWAIT = ZWAIT * w2_p$

b) Initialize shortest path algorithm:

For each access link L from origin p define

. access node N = NODAC(L)

. temporary label TIAC(N) = WWALK * ACT(L)

NOAC(N) = p

LIAC(N) = -L

c) Compute the tree of shortest paths (minimum impedance) from origin p to all network nodes:

TIAC(N) : time label at node N , N = 1,NNODET

NOAC(N) : predecessor node on path

LIAC(N) : transit line (or set of common lines) from NOAC(N) to N

and write above vectors on file TTREES

d) Complete shortest path tree by labelling all destination centroids q:

Set path time UTRP(q) = ∞

UTRP(p) = 0.0

and for each egress link L to destination centroid q define

. egress node N = NODEG(L)

. path time U = TIAC(N) + WWALK * EGT(L)

. if U $<$ UTRP(q) then define UTRP(q) = U

e) after step d is completed, vector UTRP(q) contains the value of transit impedance, u_{pq}^{tr}, between origin p and destinations q , q = 1, NCENT

f) Read in previous impedance vector OLDUP(q) , q = 1, NCENT, from random file TZIMP

g) Compare current impedance to previous one and, if different (i.e. relative difference > TOLERZ), set flag MODIF = .TRUE. and write new impedance vector for origin p, UTRP(q), on random file TZIMP

6. Reset path coefficients

WWALK = ZWALK

WWAIT = ZWAIT

NOTES : 1) Subroutine TIMPED contains two entry points:

. TIMPED: create new impedance file TZIMP

here, random file TZIMP is initialized, step f is skipped and in step g UTRP(q) is written without comparison.

.MODIMP: update and compare existing impedance file TZIMP

2) In shortest path calculations, waiting impedance is defined as follows:

WAITING TIME = MIN [MAX(WAITMI, WFACT * FLIN), WFACT * FLIN, WAITMA]

WAITING IMPEDANCE = WAITING TIME * WWAIT * $w2_p$ + WPENAL

FLIN = line headway

WFACT = regularity factor

WAITMI = minimum waiting time

WAITMA = maximum waiting time

WPENAL = waiting penalty

WWAIT * $w2_p$ = weight of waiting time

The above parameters are all user supplied.

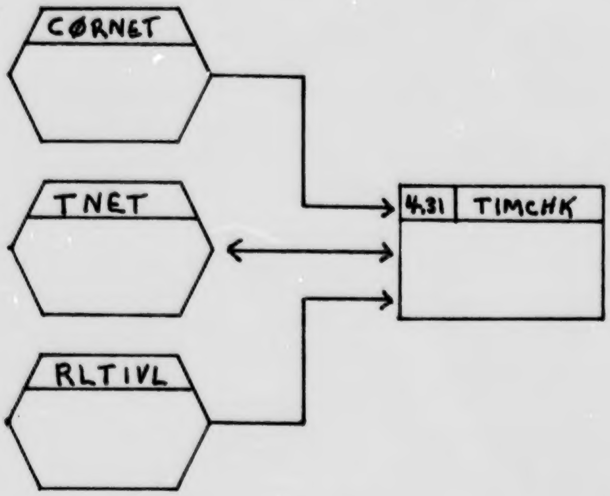
3) Two shortest path algorithms are available:

. all or nothing

. all or nothing with diversion on common line sections

MODULE : 4.31 TIMCHK

PURPOSE : Calculate new transit link times and compare



INPUT : 1. File name : CØRNET (514)

Type : Intermediate file

Symbol :

Status : Required

Contents : Vectors representing the transit line itineraries as sequences of road links: NT, NA, NR, NOT(I), NOA(J), LINKA(J), NOR(K), LINKR(K)

Sequence : Sequential by line

Source : Output from 5.14 EXPNET

2. File name : TNET (515)
Type : Intermediate file
Symbol :
Status : Required
Contents : Vectors representing the transit network structure:
FLIN, NOLIST, NONLIN, LISTN, NDIR, DA, DR, DISA, DISR, LISTL, LPNT , NADR
Sequence : Sequential
Source : Output from 5.15 TTIME ;
TIMCHK updates link time vectors DA, DR

3. File name : RLTIVL (631)
Type : Intermediate file
Symbol :
Status : Required
Contents : Predicted road link volume and travel time
Sequence : Sequential by Link L, L = 1, NLINK
Source : Output from 6.60 RASS

PROCEDURE : a) Read in the transit network vectors from file TNET:

FLIN, NOLIST, NONLIN, LISTN, NDIR, DA, DR, LISTL, LPNT, NADR

b) Read in road link travel time from file RLTIVL

c) Set MODIF = .FALSE.

d) For each line,

- read in corresponding road link sequence from file CORNET
- for each transit link of the node sequence which corresponds to a road link, calculate new bus travel time DAN (or DRN) using car time - bus time relationship and compare with previous bus time DA(I) (or DR(I)) .

  If relative difference is ≥ EPSTIM, set

  MODIF = . TRUE.

  DA(I) = DAN or DR(I) = DRN

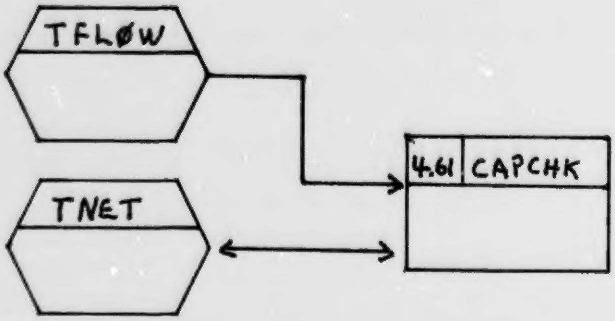
For more details on bus travel time computations see 5.15 TTIME

e) If MODIF is true, rewind file TNET and write new transit network structure:

FLIN, NOLIST, NONLIN, LISTN, NDIR, DAN, DRN, DISA, DISR, LISTL, LPNT, NADR

MODULE : 4.61 CAPCHK

PURPOSE : Adjust transit line capacity to predicted passenger volume



INPUT : 1. Data contained in COMMON areas

HDWSET(K) , K = 1, MAXHDW

is a user supplied vector of permitted headways, which is defined in a subcommand

HDWSET is sorted in ascending order by headway value

Systems default values are:

MAXHDW = 6

HDWSET = (2, 5, 10, 15, 20, 30)

2. File name : TFLØW (550)

Type : Intermediate file

Symbol :

Status : Required

Contents : Predicted passenger volumes on transit line links

Sequence : Sequential

Source : Output from 5.50 TASS

3. File name : TNET (515)

Type : Intermediate file

Symbol :

Status : Required

Contents : Vectors representing the transit network structure:

FLIN, NOLIST, NONLIN, LISTN, NDIR, DA, DR, DISA, DISR, LISTL, LPNT, NADR

Sequence : Sequential

Source : Output from 5.15 TTIME;

CAPCHK updates headway vector FLIN

PROCEDURE : 1) Read in the transit network vectors from file TNET:

FLIN, NOLIST, NONLIN, LISTN, NDIR, DA, DR, DISA, DISR, LISTL, LPNT, NADR

2) Set MODIF = .FALSE.

3) For each line L

a) Read in link volumes for line L and determine FLOMAX, the maximum passenger flow on line L

b) Select the headway HNEW, in set HDWSET, which is just sufficient to provide adequate capacity

c) If the previous headway FLIN(L) is different from HNEW define the new headway

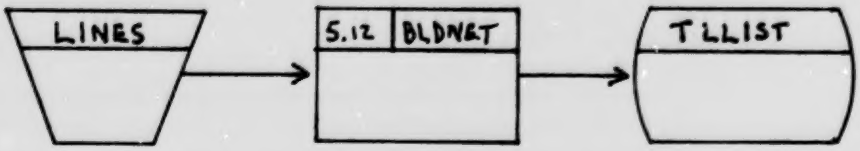
FLIN(L) = HNEW and set MODIF = .TRUE.

4) If MODIF is true, rewind file TNET and write new transit network structure:

FLIN, NOLIST, NONLIN, LISTN, NDIR, DA, DR, DISA, DISR, LISTL, LPNT, NADR

MODULE : 5.12 BLDNET

PURPOSE : Read in transit line descriptions and construct transit network (tables and lists)



INPUT : File name : LINES (201)

Type : Data base

Symbol :

Status : Required

Contents : Data describing each transit line

first record: line characteristics

a) essential data

- Line number
- Line name
- Headway

b) optional data

- Total length of itinerary
- Mean speed (commercial speed)
- Line type
- Layover time
- Number of vehicles

following records: line itinerary

- sequence of directed nodes

Sequence : Sequential

Source : User specified

Note : Lines are assigned an internal line number (1 to NLINE) by the data base service modules

OUTPUT : File name : TLLIST (501)

Type : Stored in common area /STRUC/

Symbol :

Status : Always created

Contents : Vectors representing the transit network structure:

- FLIN(L) : headway of line L (L = 1, NLINE)
- NOLIST(L) : pointer to the beginning of the itinerary of line L in vector LISTN (L = 1, NLINE)
- NONLIN(L) : pointer to the end of the itinerary of line L in vector LISTN (L = 1, NLINE)
- LISTN(J) : node sequence on the itinerary of line L (J = NOLIST(L), NONLIN(L)) (L = 1, NLINE)
- NDIR(J) : node direction for LISTN(J)
- $\left\{\begin{matrix}\text{LISTL(K)}\\ \text{LPNT(K)}\\ \text{NADR(K)}\end{matrix}\right\}$: list of lines adjacent to each node

Use : 1) To generate file CØRNET in 5.14 EXPNET

2) Transferred to file TNET in 5.15 TTIME

PROCEDURE : a) For each line

- Read in the line headway H and the directed node sequence from file LINES
- Store H in FLIN(L)
- Store the node numbers in vector LISTN and direction in vector NDIR (blank = 0, A = + 1, R = -1)
- Define pointers NOLIST(L) : address of starting terminus in LISTN

  NONLIN(L) : address of ending terminus in LISTN
- Redefine node numbers : LISTIN(J) = LISTN(J) - NSPENO*
- For each node in the sequence, insert an entry in the list of lines adjacent to that node (i.e. vectors LISTL(K), LPNT(K) and NADR(K))

  For $K \leq$ NNODET,

  LISTL(K) contains the number of the first line adjacent to node K

  LPNT(K) contains a pointer J in LISTL to another line adjacent to node K ($J >$ NNODET)

  NADR(K) contains the address of node K in LISTN

  For $K >$ NNODET,

  vectors LISTL, LPNT and NADR contain the rest of the list of lines adjacent to nodes

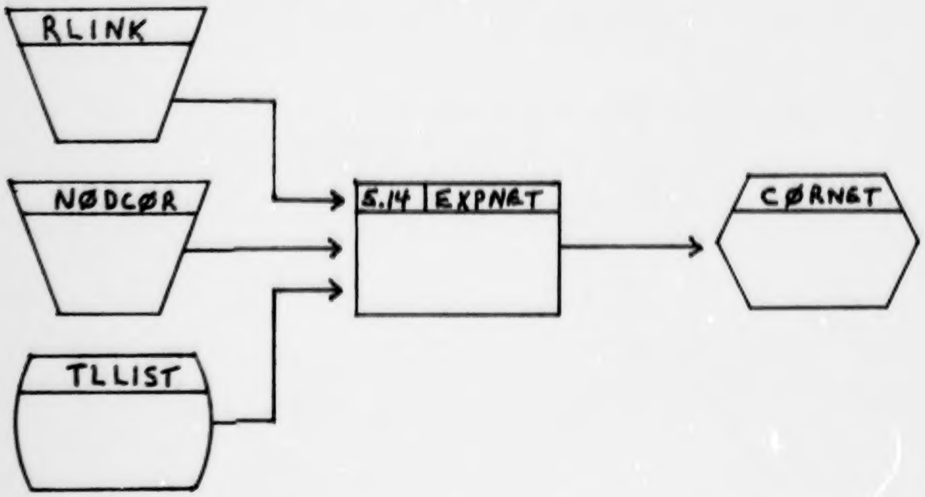
NOTE: for each node, the list of lines adjacent is sorted by increasing headway value

* See "INTERNAL NODE NUMBERS"

b) For validation, print out the list of lines passing at each node and also the list of all isolated nodes.

MODULE : 5.14 EXPNET

PURPOSE : Construct an expanded transit network representation to permit interface with the road network



INPUT :	1. File name	:	RLINK (211)
	Type	:	Data base
	Symbol	:	
	Status	:	Required
	Contents	:	Data describing each link in the road network
	Sequence	:	Sequential in 2 sections; only section 2 (links between nodes) is used in the procedure
	Source	:	User specified

2. File name : NØDCØR (202)
Type : Data base
Symbol :
Status : Required
Contents : Transit node - road nodes correspondence table
Sequence : Sequential and sorted by the data base service modules in ascending order by transit node
Source : User specified

3. File name : TLLIST (501)
Type : Stored in common area /STRUC/
Symbol :
Status : Required
Contents : Vectors representing the transit network structure:
FLIN(L), NOLIST(L), NONLIN(L),
LISTN(J), NDIR(J),
LISTL(K), LPNT(K) , NADR(K)
only NOLIST, NONLIN, LISTN and NDIR are used in the procedure
Source : Created in 5.12 BLDNET

OUTPUT : File name : CØRNET (514)
Type : Intermediate file
Symbol :
Status : Always created

Contents	:	Vectors representing the transit line itineraries as sequences of road links, including turning links

For each line, the following information is written,

NT : number of transit-only nodes removed

NA : number of nodes on forward itinerary

NR : number of nodes on reverse itinerary

NOT(I),I = 1,NT : list of transit-only nodes removed

NOA(I),I =1,NA : node sequence on forward itinerary

LINKA(I),I = 2,NA: road link numbers for above sequence

NOR(I),I = 1,NR : node sequence on reverse itinerary

LINKR(I),I = 2,NR: road link numbers for above sequence

Sequence : Sequential by line

Use : 1) Determine transit link time from road link time or road link length in 5.15 TTIME

2) Calculate bus frequency by road link in 4.18 BUSFRQ

3) Determine new transit link time from predicted road link time in 4.31 TIMCHK

PROCEDURE : 1) Read in the list of road links from section 2 of file RLINK ignoring egress links

2) Read in the transit node-road nodes correspondence table from file N0DC0R

3) Read in vectors NOLIST, NONLIN, LISTN and NDIR from file TLLIST

4) For each line, construct a new node sequence:

   a) Consider the forward itinerary node sequence in LISTN and define a new node sequence in NOA according to the following rules:

   - a road node remains unchanged
   - a transit node not corresponding to any road node remains unchanged
   - a transit node corresponding to a zero node is removed from the sequence and stored in vector NOT
   - a transit node corresponding to a single road node is replaced by that road node
   - a transit node corresponding to more than one road node (expanded intersection) is replaced by
     - . one of the road nodes, if the transit node is a terminus node
     - . two of the road nodes, otherwise

   b) Store in vector LINKA the numbers of the road links defined by the node sequence in NOA

For the link NOA(I - 1), NOA(I)

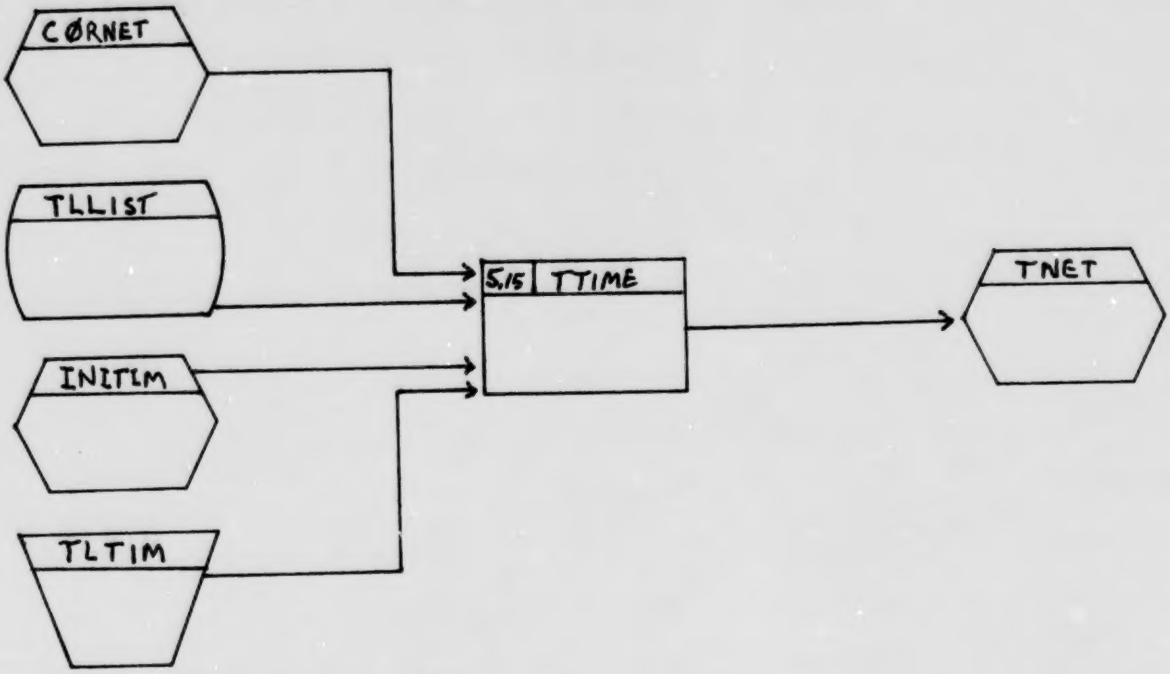
$$\text{LINKA(I)*} = \begin{cases} 0 & \text{if the link is a transit-only link} \\ -1 & \text{if the link is a transit-only turning link} \\ + \text{ (link number)} & \text{if the link is a regular road link} \\ - \text{ (link number)} & \text{if the link is a turning road link} \end{cases}$$

c) Repeat steps a and b for the reverse itinerary, using vectors NOR and LINKR

* A negative number in LINKA indicates a link crossing an expanded intersection

MODULE : 5.15 TTIME

PURPOSE : Define initial travel time on the links of the transit network
Define link lengths on the transit network



INPUT : 1. File name : CØRNET (514)
Type : Intermediate file
Symbol :
Status : Required
Contents : Vectors representing the transit line itineraries as sequences of road links:
NT, NA, NR,
NOT(I), NOA(J), LINKA(J), NOR(K), LINKR(K)
Sequence : Sequential by line
Source : Output from 5.14 EXPNET

2. File name : TLLIST (501)
Type : Stored in common area /STRUC/
Symbol :
Status : Required
Contents : Vectors representing the transit network structure:
FLIN(L), NOLIST(L), NONLIN(L),
LISTN(J), NDIR(J),
LISTL(K), LPNT(K), NADR(K)
Source : Created in 5.12 BLDNET

3. File name TLTIM (204)
Type : Data base
Symbol :
Status : Optional (see PROCEDURE)
Contents : User specified travel time for transit-only links
Sequence : Sequential and sorted by the data base service modules in ascending order by link origin and destination node
Source : User specified

4. File name : INITIM (205)
Type : Intermediate file
Symbol :
Status : Required

Contents : Data used to determine initial transit times on transit links corresponding to road links

One record for each road link:

- RLL: road link length from file RLINK
- RLT: car travel time { from file RLINK if observed / from an old RLTIVL if predicted }

Sequence : Sequential by link. The sequence is the same as the RLINK sequence

Source : RLINK is user specified and RLTIVL has been created in a previous road assignment run.

OUTPUT : File name : TNET (515)

Type : Intermediate file

Symbol : Always created

Contents : Vectors representing the transit network structure and travel time on the transit network:

FLIN, NOLIST, NONLIN, LISTN, NDIR, LISTL, LPNT, NADR are as defined for file TLLIST (see module 5.12)

DA(J) : travel time on forward itinerary
DISA(J): link length on forward itinerary
DR(J) : travel time on reverse itinerary
DISR(J): link length on reverse itinerary

Sequence : Sequential

Use : 1) To calculate bus frequency in 5.18 BUSFRQ

2) To calculate shortest paths in 4.21 TIMPED and in 5.50 TASS

3) To compare link travel times in 4.31 TIMCHK

4) To compare line capacity with predicted volume in 4.61 CAPCHK

Updated : 1) Travel times are modified by 4.31 TIMCHK

2) Headways are modified by 4.61 CAPCHK

PROCEDURE : a) Read in transit network vectors from file TLLIST

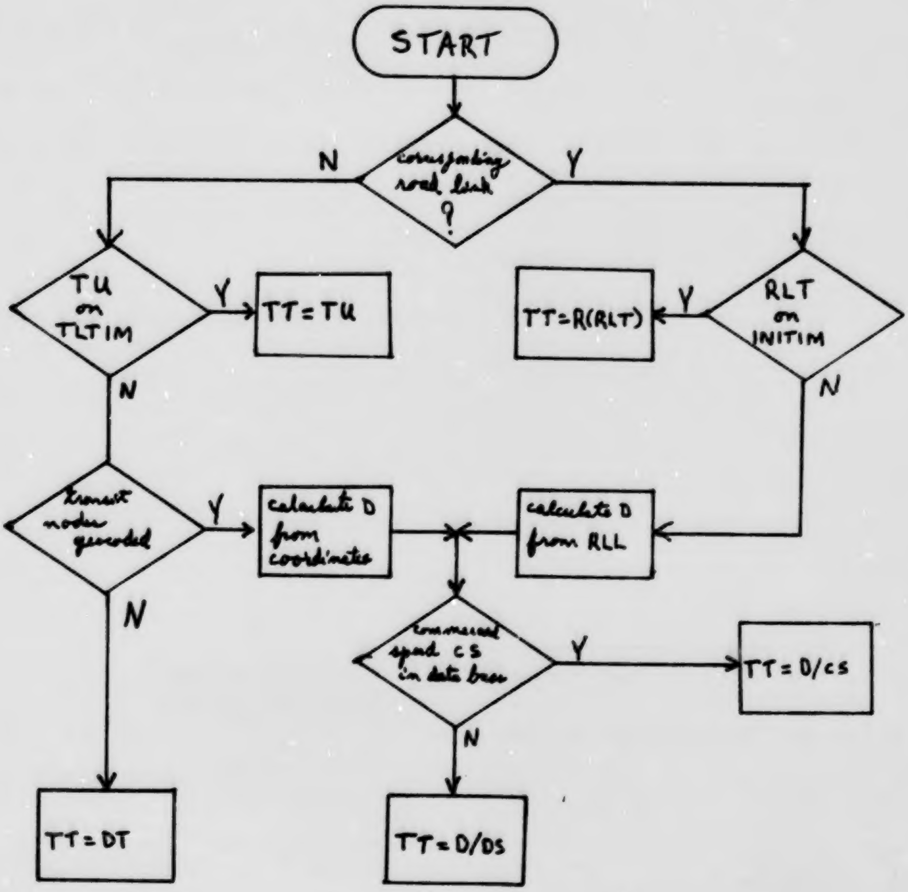
b) Read in road link data from file INITIM

c) If a TLTIM file is supplied, read in travel time for transit-only links

d) Initialize link lengths to "undetermined" (=99.)

e) For each line, define in DA, DR the transit link times corresponding to the node sequence:

- read in corresponding road link sequence from file C0RNET
- for each transit link of the node sequence, in LISTN, calculate travel time TT using the following algorithm:


START
corresponding road link ?
N
Y
TU on TLTIM
Y
TT=TU
N
RLT on INITIM
Y
TT=R(RLT)
N
transit nodes geocoded
Y
calculate D from coordinates
N
calculate D from RLL
commercial speed CS in data base
Y
TT=D/CS
N
TT=DT
TT=D/DS

where,

TU : user supplied travel time (file TLTIM)

RLT : car travel time (file INITIM)

R : a function relating bus speed to car speed *

RLL : road link length (file INITIM)

D : length of the transit link

CS : commercial speed of the line (from Data Base)

DS : system's default value for commercial speed, which may be modified in a subcommand

DT : system's default value for transit link time, which may be modified in a subcommand

f) While performing step e , every time there is a one-to-one correspondence between a transit link and a road link, set transit link length (DISA or DISR) equal to the road link length.

g) For all other transit links, determine the distance from node coordinates

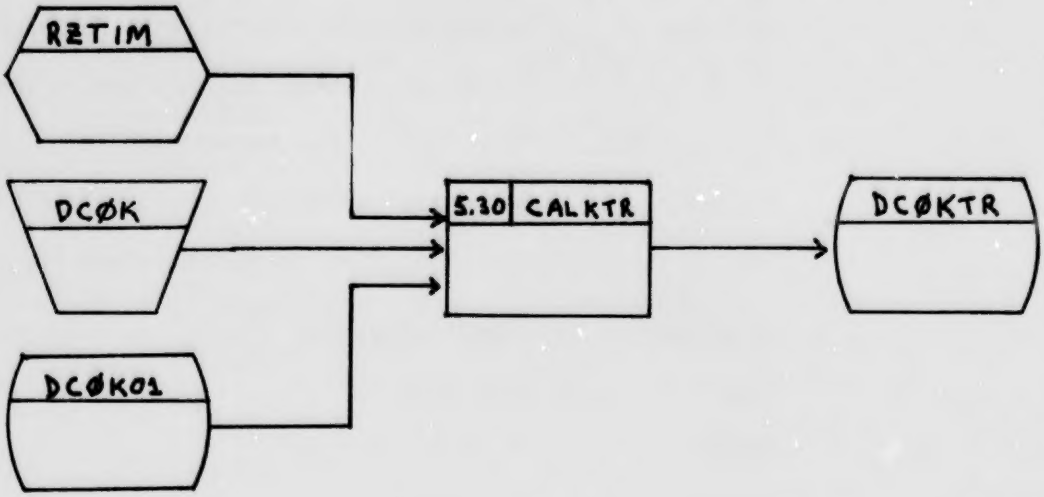
h) Write vectors
FLIN, NOLIST, NONLIN, LISTN, NDIR, DA, DR, DISA, DISR, LISTL, LPNT, NADR on file TNET

---

* R is specified by two subprograms: Function BUSROD
Function BUSCAR
which may be modified by the user.

MODULE : 5.30 CALKTR

PURPOSE : Calculate the component of the demand function which is independent of the transit impedance u_{pq}^{tr}



INPUT : 1. File name : RZTIM (632)

Type : Intermediate file

Symbol : T_{pq}^{au}

Contents : Vector of interzone travel times by shortest path for automobile, T_{pq}^{au}, from origin p to destinations q , q = 1, NCENT

Sequence : Sequential by origin p , p = 1, NCENT

Source : External user supplied file which may have been obtained from a previous traffic assignment (output from 6.60 RASS)

2. File name : DC0K (231)
Type : Data base
Symbol : $k0_p$, $k2_p$, $k3_p$
Status : Required
Contents : Demand function coefficients for each origin p , where $k2_p$ is the in-vehicle time coefficient for automobile
Sequence : Sequential by origin p , p = 1, NCENT
Source : User specified, computed externally

3. File name : DC0K01 (702)
Type : Work file
Symbol : $k0_p + \sum c_i x_i = k01_{pq}$
Status : Required
Contents : Component of demand function which is independent of transit and automobile impedances. 1 record for each origin p containing the values of $k01_{pq}$ to all destinations q , q = 1, NCENT
Sequence : Sequential by origin p , p = 1, NCENT
Source : Created by 7.10 DC0S01

OUTPUT : File name : DC0KTR (511)
Type : Work file
Symbol : K^{tr}_{pq}
Status : Always created

Contents : Component of demand function which is independent of transit impedance. One record for each origin p containing the demand function component K_{pq}^{tr} to all destinations q , $q = 1$, NCENT

Sequence : Sequential by origin p , $p = 1$, NCENT

Use : Input to 5.50 TASS

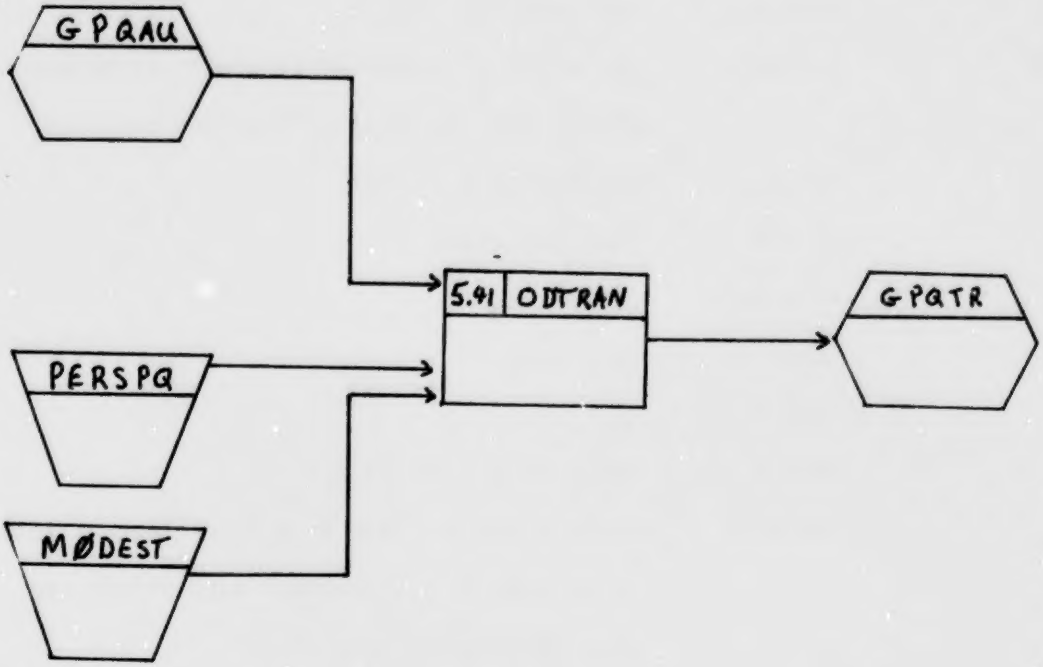
PROCEDURE : The program functions are as follows:

For each origin p , $p = 1$, NCENT

- Read in-vehicle time coefficient $k2_p$ from file DC0K
- Read demand function component $k01_{pq} = k0_{pq} + \sum c_i \; x_i$ for all destinations q , $q = 1$, NCENT from file DC0K01
- Read automobile interzone times T^{au}_{pq} for all destinations q , $q = 1$, NCENT from file RZTIM
- Calculate $K^{tr}_{pq} = k01_{pq} - k2_p \; T^{au}_{pq}$ for all q , $q = 1$, NCENT
- Write demand function component K^{tr}_{pq} , $q = 1$, NCENT on file DC0KTR

MODULE : 5.41 ODTRAN

PURPOSE : Calculate current transit demand for fixed demand transit assignment



INPUT : 1. File name : GPQAU (633)

Type : Intermediate file

Symbol : g_{pq}^{au}

Status : Required for option 1

Contents : Vector of predicted person trips using auto mode g_{pq}^{au} from origin p to destinations q , q = 1, NCENT

Sequence : Sequential by origin p , p = 1, NCENT

Source : Output from 6.60 RASS

2. File name : PERSPQ (221)

Type : Data base

Symbol : $\bar{g}_{pq}$

Status : Required

Contents : OD matrix of number of person trips between each origin and each destination centroid

Sequence : Sequential by origin

Source : User specified

3. File name : MØDEST (223)

Type : Data base

Symbol : μ_{pq}

Status : Required for option 2 a

Contents : Modal split estimate as a percentage of total trips made by auto between each origin and each destination centroid

Sequence : Sequential by origin

Source : User specified

OUTPUT : File name : GPQTR (541)

Type : Intermediate file

Symbol : g_{pq}^{tr}

Status : Always created

Contents : Vector of transit trips g_{pq}^{tr} from origin p to destinations q , q = 1, NCENT (packed OD matrix format)

Sequence : Sequential by origin p , p = 1, NCENT

Use : OD matrix for 5.50 TASS (transit assignment with fixed demand)

PROCEDURE:

Option 1: Calculate transit trips from auto trips

a) $g_{pq}^{tr} = \bar{g}_{pq} - g_{pq}^{au}$ predicted auto trips

b) $g_{pq}^{tr} = \bar{g}_{pq} - \mu_{pq}$ user-estimated auto trips

$\bar{g}_{pq}$ = person trips by all modes

g_{pq}^{au} = person trips by car predicted from previous assignment

μ_{pq} = estimated person trips by car (user-supplied)

Option 2: Calculate transit trips from modal split estimate:

a) $g_{pq}^{tr} = \bar{g}_{pq} * [1-\mu_{pq}]$ variable modal split

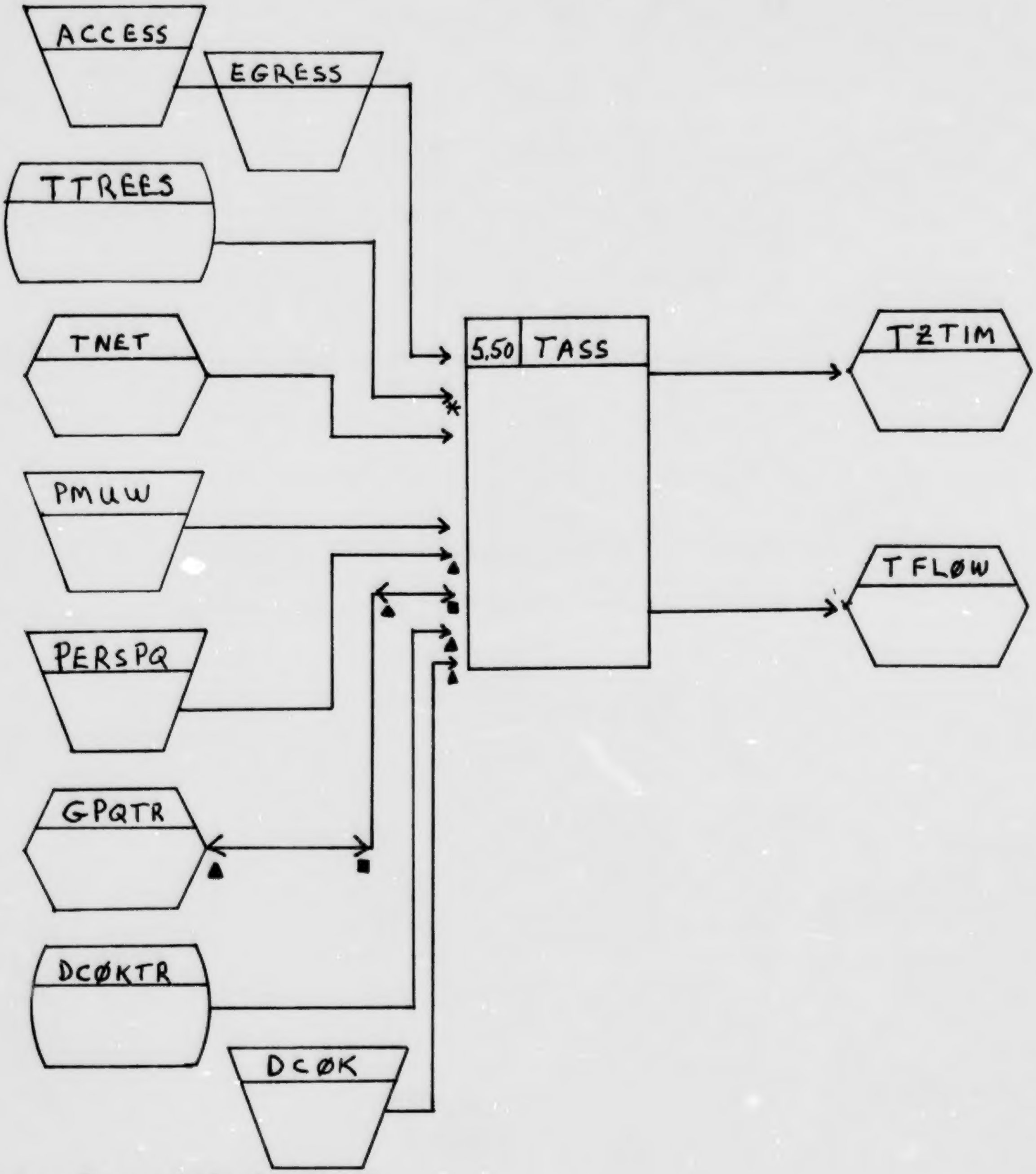
b) $g_{pq}^{tr} = \bar{g}_{pq} * [1-\mu]$ fixed modal split

NOTE : For bimodal assignment, option 1 a) is always used.

For all other cases, default option is 2 b) with $\mu = 0.0$

MODULE : 5.50 TASS

PURPOSE : Perform a transit assignment with variable or fixed demand


ACCESS
EGRESS
TTREES
TNET
PMUW
PERSPQ
GPQTR
DCØKTR
DCØK
5.50
TASS
TZTIM
TFLØW
*
▲
■

▲ : For variable demand only

■ : For fixed demand only

*: For bimodal assignment only

File	Contents	Sequence	Source	Status
INPUT FILES				
ACCESS (203)	Access links from zone centroids to transit network nodes and corresponding access time	Sequential, sorted in ascending order by zone centroid	User specified	Required
EGRESS (203)	Egress links from transit network nodes to zone centroids and corresponding egress time	Sequential, sorted in ascending order by zone centroid	User specified	Required
TNET (515)	Vectors representing the transit network structure	Sequential	Output from 5.15 TTIME	Required
DCØK (231)	Demand function coefficients for each origin p : $k0_p$, $k2_p$, $k3_p$	Sequential by origin p , $p = 1$, NCENT	Computed externally	Required for variable demand only
GPQTR (541)	Vector of transit trips g^{tr}_{pq} from origin p to destinations q , $q = 1$, NCENT	Sequential by origin p , $p = 1$, NCENT	Output from 5.41 ODTRAN	Required for fixed demand only
PERSPQ (221)	OD matrix of number of person trips between each origin and each destination, $\bar{g}_{pq}$	Sequential by origin	User specified	Required for variable demand only
DCØKTR (511)	Vector of demand function components K^{tr}_{pq} for origin p and destinations q , $q = 1$, NCENT	Sequential by origin p , $p = 1$, NCENT	Output from 5.30 CALKTR	Required for variable demand only
PMUW (250)	Transit path coefficients multipliers $w1_p$, $w2_p$	Sequential by origin p , $p = 1$, NCENT	User specified	Optional

File	Contents	Sequence	Source/Use	Status
TTREES (521)	Shortest path tree from each origin to all network nodes	Sequential	Source: module 4.21 TIMPED	
OUTPUT FILES				
GPQTR (541)	Vector of predicted transit trips g_{pq}^{tr} from origin p to destinations q , q = 1, NCENT	Sequential by origin p , p = 1, NCENT	For reports	Created for variable demand only
TZTIM (532)	Interzone transit travel time: $T_{pq}^{tr}(v)$: in-vehicle time $T_{pq}^{tr}(acc)$: access and egress time $T_{pq}^{tr}(w)$: waiting time N_{pq}^{tr} : number of line sections for all q , q = 1, NCENT	Sequential by origin p , p = 1, NCENT	For reports	Always created
TFLØW (550)	Transit network predicter passenger volumes	Sequential	1. Input to 4.61 CAPCHK 2. For reports	Always created

PROCEDURE : 1) Read in the transit network from file TNET:

NLINE, NSEG, NSEL and vectors

FLIN, NOLIST, NONLIN, LISTN, NDIR, DA, DR, LISTL, LPNT, NADR

2) Read in access links from file ACCESS and construct the following vectors:

NAL = number of access links

NAC(p) = number of first access link from centroid p, p = 1, NCENT

NAC(NCENT + 1) = NAL + 1

NODAC(L) = destination node of access link L, L = 1, NAL

ACT(L) = access time on access link L, L = 1, NAL

3) Read in egress links from file EGRESS and construct the following vectors:

NEL = number of egress links

NEG(p) = number of first egress link to centroid p p = 1, NCENT

NEG(NCENT + 1) = NEL + 1

NODEG(L) = origin node of egress link L, L = 1, NEL

EGT(L) = egress time on egress link L, L = 1, NEL

4) Save path coefficients

ZWALK = WWALK

ZWAIT = WWAIT

5) For each origin p

A. If path multipliers w are to be used, read $w1_p$, $w2_p$ from file PMUW and define path coefficients for origin p

$WWALK = ZWALK * W1_p$

$WWAIT = ZWAIT * w2_p$

```
B. Initialize shortest path algorithm:
   For each access link L from origin p define
   . access node N = NODAC(L)
   . temporary label   $TIAC(N) = WWALK * ACT(L)$
                       $NOAC(N) = p$
                       $LIAC(N) = -L$
C. Compute the tree of shortest paths (minimum impedance)
   from origin p to all network nodes:
   TIAC(N):  time label at node N , N = 1, NNODET
   NOAC(N):  predecessor node on path
   LIAC(N):  transit line (or set of common lines)
             from NOAC(N) to N
D. For all destinations  q , q = 1, NCENT
   a) Complete shortest path from origin p to destination q
      - set path impedance $UTRP(q) = \infty$
      - set egress node    $NEGRES = 0$
      - for each egress link  L  to destination centroid
        q define:
        . egress node:  $N = NODEG(L)$
        . impedance:  $U = TIAC(N) + WWALK * EGT(L)$
        . if $U < UTRP(q)$ then define  $\left[ \begin{array}{l} UTRP(q) = U \\ NEGRES = N \end{array} \right.$
   b) Determine demand  g  for OD pair  p,q:
```

$$g = \begin{cases} g_{pq}^{tr} & \text{for fixed demand} \\ & \\ f(\bar{g}_{pq}\ ,\ K_{pq}^{tr}\ ,\ UTRP(q)) & \text{for variable demand} \end{cases}$$

where f is the demand (modal split) function

c) Accumulate trips, g , on path pq

. flow on egress link in vector EGFLO

. flow on transit network in vectors DY and DZ which correspond to the node sequences

. flow on access link in vector ACFLO

and accumulate path time components

. access time: TACC(q) = ACT (access link)
+ EGT (egress link)

. waiting time: TWAIT(q) = sum of waiting times on path

. in-vehicle time: TVEH(q) = sum of travel times on path

. line sections: NSEC(q) = number of line sections on path

E. Save path time components for origin p on file TZTIM:

TACC(q) , q = 1, NCENT

TWAIT(q), q = 1, NCENT

TVEH(q) , q = 1, NCENT

NSEC(q) , q = 1, NCENT

6) Reset path coefficients

WWALK = ZWALK

WWAIT = ZWAIT

7) Print out predicted volumes

8) Save predicted volumes on file TFLOW

NOTES: 1) Notes 2 and 3 in the description of module 4.21 TIMPED also apply here.

2) The procedure to determine transit demand (step 5.D.b) will now be outlined

3) In the case of bimodal assignment, step 5.C becomes: Read vectors TIAC, NOAC, LIAC from file TTREES

PROCEDURE (to determine transit demand, module 5.55 TDEM) :

- for fixed demand only,

  read in vector of transit trips g_{pq}^{tr} to all destinations q , $q = 1$, NCENT from file GPQTR

- for variable demand only,

  - read in vector of total person trips $\bar{g}_{pq}$ to all destinations q , $q = 1$, NCENT from file PERSPQ
  - read in vector of demand function component K_{pq}^{tr} from file DC0KTR
  - read in demand coefficient $k3_p$ from file DC0K
  - for each destination q , $q = 1$, NCENT calculate:

    $U = U_{pq}^{tr} * k3_p$

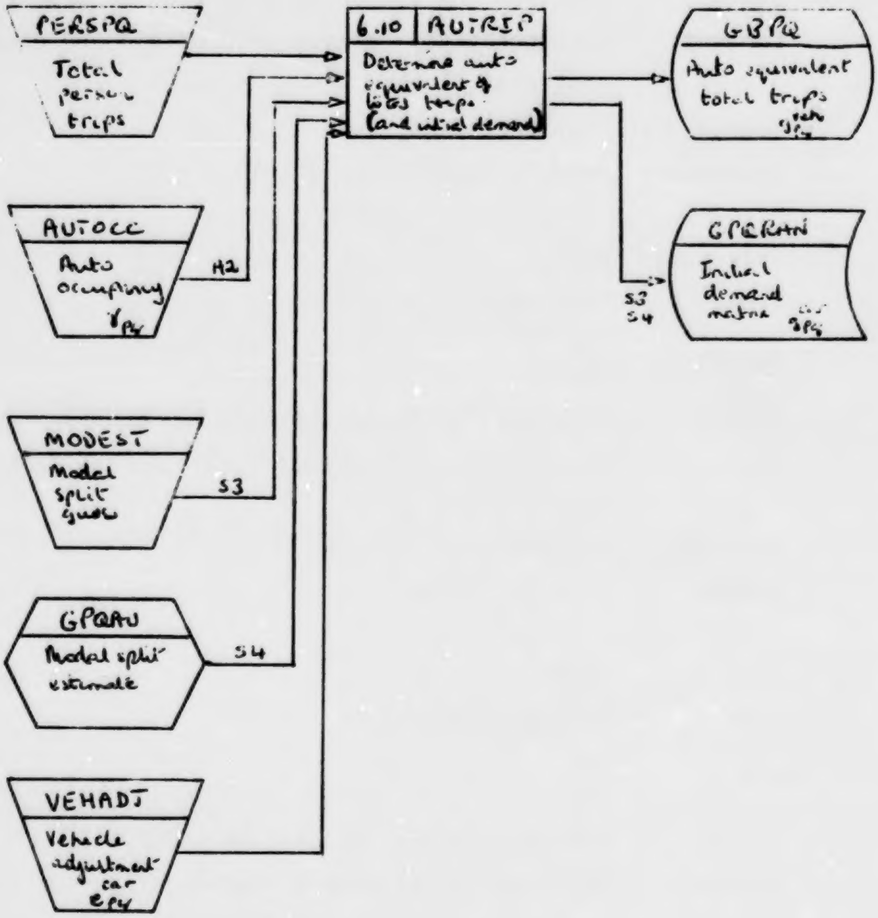
    $g_{pq}^{tr} = \bar{g}_{pq} * (1.0 - K_{pq}^{tr} - U)$ linear model

    or

    $g_{pq}^{tr} = \bar{g}_{pq} * [1.0 - 1.0 / EXP(K_{pq}^{tr} + U)]$ logit model

MODULE: 6.10 AUTRIP

PURPOSE: Determine auto equivalent of total trips


PERSPQ
Total person trips
6.10
AUTRIP
Determine auto equivalent of total trips (and initial demand)
GBPQ
Auto equivalent total trips
AUTOCC
Auto occupancy
H2
GPERAN
Initial demand matrix
S3
S4
MODEST
Modal split guess
S3
GPQAU
Modal split estimate
S4
VEHADJ
Vehicle adjustment car

INPUT 1. File name: PERSPQ

Type : Data base

Symbol : $\bar{g}_{pq}$

Status : Required

Contents : OD matrix of number of person trips between each origin and each destination centroid.

Source : User specified

Sequence : Read in sequence by origin

2. File name: AUTOCC
Type : Data base
Symbol : γ_{pq}
Status : Required for Option A2 only (see PROCEDURE)
Contents : Auto occupancy between each origin and each destination centroid
Source : User specified
Sequence : Read in sequence by origin

3. File name: MODEST
Type : Data base
Symbol : g^{au}_{pq}
Status : Required for Option S3 (see PROCEDURE)
Contents : Modal split estimate as number person trips between each destination g^{au}_{pq}
Sequence : Sequential by origin
Source : User specified

4. File name: GPQAU
Type : Intermediate file
Symbol : g^{au}_{pq}
Status : Required for Option S4 (see PROCEDURE)
Contents : Modal split estimate as number of person trips between each origin and each destination, g^{au}_{pq}
Sequence : Sequential by origin
Source : Output from 6.80 RASSAV

5. File name: VEHADJ
Type : Data base
Symbol : e^{car}_{pq}
Status : Required if variable ODEXSW in COMMON/APARM/ is TRUE (Subcommand parameter VEAD=YES)
Contents : Vehicle adjustment matrix as number of vehicles to be added to those required for person trips between each origin and each destination
Sequence : Sequential by origin
Source : User specified.

OUTPUT 1. File name: GBPQ

Type : Work file

Symbol : $\bar{g}_{pq}^{veh}$

Status : Always created

Contents : OD matrix of auto (vehicle) trips equivalent to OD matrix of total person trips

Sequence : Written in sequence by origin for all destinations

Use : a) OD matrix in road traffic assignment with fixed demand

or b) To create initial auto demand matrix for variable demand traffic assignment

2. File name: GPQRAN

Type : Work file

Symbol : g_{pq}^{car}

Status : Created for Option S3 and S4 only

Contents : Matrix of auto trips between each origin and all destinations

Sequence : Random, indexed by origin

Use : Initial value of demand in traffic assignment

PROCEDURE: 1. Calculate the vehicle equivalent of total person trips

Option A1 $\bar{g}_{pq}^{veh} = \bar{g}_{pq} / \gamma$

Option A2 $\bar{g}_{pq}^{veh} = \bar{g}_{pq} / \gamma_{pq}$ q = all destinations

The user specifies, in a subcommand, which option he requires.

Option A1 assumes a fixed auto occupancy for all origins and destinations. The user may specify the occupancy in the subcommand. If no value is specified the system default value is used (system generated)

Option A2 allows the auto occupancy to vary with origin and destination and file AUTOCC is required.

2. Calculate the vehicle equivalent of the initial auto demand matrix if this has been specified in persons (Options S3 and S4). If it has been specified as a percentage of total trips or as the number of automobile trips the matrix is created in module 6.15 INITDM.

Options S3 or S4,
& option A1 $\quad g_{pq}^{car} = (g_{pq}^{au} \; / \; \gamma) + e_{pq}^{car}$

Options S3 or S4
with option A2 $\quad g_{pq}^{car} = (g_{pq}^{au} \; / \; \gamma) + e_{pq}^{car} \qquad q = \text{all destinations}$

The user specifies in a subcommand which options he requires

Option S3 assumes that the number of person trips by auto is on file MODEST

Option S4 assumes that the number of person trips by auto is on file GPQAU

e_{pq}^{car} is assumed to be zero if ODEXSW in /APARM/ is FALSE. (Subcommand parameter VEAD = NO)

The program functions are as follows:

For each origin p ; p = 1, NCENT:

- Read person trips to all destinations q from file PERSPQ
- (ODEXSW = TRUE) Read vehicle adjustment to all destinations from file VEHADJ
- (Option A2 only) Read auto occupancy to all destinations q from file AUTOCC
- (Option S3 only) Read number of person trips by auto to all destinations q from file MODEST
- (Option S4 only) Read number of person trips by auto to all destinations q from file GPQAU
- Calculate vehicle tripa $\bar{g}_{pq}^{veh}$ according to the option (A1 or A2)
- Options S3 and S4 only) Calculate initial auto demand according to the options (A1 or A2, and S3 or S4)
- Write vehicle trips to all destinations on file GBPQ
- (Options S3 and S4 only) Write auto trips from origins as indexed record (by origin) on file GPQRAN

The program functions are summarized in tabular form at the end of the module description.

Files PERSPQ (AUTOCC for Option A2, MODEST for Option S3, and GPQAN for Option S4) are assumed to be "sorted" in ascending order by origin.

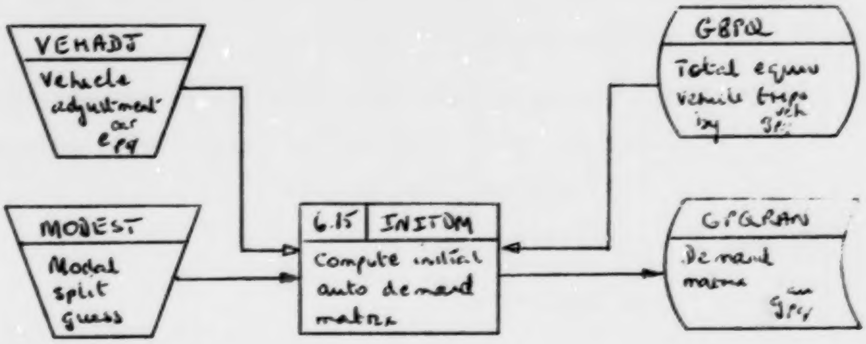
For Option A2, if $\gamma_{pq} = 0$ for a pq for which $\overline{g}_{pq}$ is non zero the system default value will be used for γ, and a message will be printed.

6.10 AUTRIP Program functions for different Options

Program functions	AOCC		SPLT		
	xx.x	PERS	PERS	PRED	VALU, PCNT, AUTO
	A1	A2	S3	S4	S1, S2, S5
Rewind AUTOCC		x			
Open GPQRAN			x	x	
Rewind MODEST			x		
Rewind GPQAU				x	
Read OD matrix	x	x			
Read AUTOCC		x			
Read MODEST			x		
Read GPQAU				x	
Calc. GBPQ=Total/γ	x	x			
Calc. GPQCAR=Splt/γ			x	x	
Write GBPQ	x	x			
Write GPQRAN			x	x	
End file GBPQ	x	x			
Rewind GPQAU				x	
Force EOF,AUTOCC		x			
Force EOF,MODEST			x		

MODULE: 6.15 INITDM

PURPOSE: Create initial auto demand matrix


VEHADJ
Vehicle adjustment e^{car}_{pq}
MODEST
Modal split guess
6.15 INITDM
Compute initial auto demand matrix
GBPQ
Total equiv vehicle trips $\bar{g}^{veh}_{pq}$
GPQRAN
Demand matrix g^{au}_{pq}

INPUT 1. File name: GBPQ

Type : Work file

Symbol : $\bar{g}^{veh}_{pq}$

Status : Required

Contents : OD matrix of auto (vehicle) trips equivalent to OD matrix of total person trips

Sequence : Read in sequence by origin

Source : Output from 6.10 AUTRIP

2. File name: MODEST

Type : Data base

Symbol : m_{pq} or m or g^{au}_{pq}

Status : Required for Options S1, S2, S5 (see PROCEDURES)

Contents : Modal split estimate as a percentage or number of trips.

- (Option S1) Percentage of total trips made by auto between all origins and destinations centroids, m.
- (Option S2) Percentage of total trips made by auto between each origin and each destination centroid, m_{pq} .
- (Option S5) Estimate of the number of auto trips between each origin and each destination centroid, g^{au}_{pq}

Sequence : Sequential by origin (Options S2 and S5)

Source : User specified

3. File name: VEHADJ

Type : Data base

Symbol : e_{pq}^{car}

Status : Required if variable ODEXSW in COMMON/APARM/ is TRUE (Subcommand parameter VEAD = YES)

Contents : Vehicle adjustment matrix as number of vehicles to be added to those required for person trips between each origin and each destination.

Sequence : Sequential by origin

Source : User specified.

OUTPUT: File name: GPQRAN
Type : Work file
Symbol : g_{pq}^{car}
Status : Always created
Contents : Matrix of auto trips between each origin and all destinations.
Sequence : Random, indexed by origin
Use : Initial value of demand in traffic assignment with variable demand.

PROCEDURE: (For option S1, the percentage of total trips made by auto is the same for all origins and destinations and is specified in a subcommand initially if the user specifies a value, otherwise the system default = 1.0 is used. Options S3 and S4 are handled in 6.10 AUTRIP). The program functions are summarized in tabular form following this description.

For each origin

- (Options S1, S2) Read in total equivalent vehicle trips from file GBPQ
- (Option S2) Read in percentage of auto trips from file MODEST
- (Options S1, S2) Calculate auto trips

  $g_{pq}^{car} = \bar{g}_{pq}^{veh} * m$ Option S1

  $g_{pq}^{car} = \bar{g}_{pq}^{veh} * m_{pq}$ Option S2
- (Option S5) Read in estimate of number of auto trips from file MODEST
- Add vehicle adjustment

  $g_{pq}^{car} = g_{pq}^{car} + e_{pq}^{car}$ q = all destinations

  (e_{pq}^{car} is assumed to be zero if ODEXSW in /APARM/ is FALSE i.e. subcommand parameter VEAD = NO).
- Write auto trips for origin as indexed record (by origin) on file GPQRAN.

6.15 INITDM Program functions for different options

Program functions	SPLT VALU S1	 PCNT S2	 AUTO S5	 PERS,PRED S3, S4
Open GPQRAN	x	x	x	
Rewind MODEST		x	x	
Read GBPQ	x	x		
Read MODEST=μ		x		
Calc. GPQCAR=GBPQ*μ	x	x		
Read MODEST=GPQCAR			x	
Write GPQRAN	x	x	x	
Force EOF,MODEST		x	x	
Close GPQRAN	x	x	x	

MODULE : 6.20 TRAFIC

PURPOSE : Set up data and perform variable or fixed demand road traffic assignment

INPUT : a) Data contained in COMMON area /APARM/

LOGICAL RVARSW	TRUE indicates variable demand assignment, otherwise fixed demand.
LOGICAL BIMOSW	TRUE indicates bimodal assignment.
LOGICAL BUSISW	TRUE indicates bus frequency or road links to be used.

b) Subroutine parameter

INTEGER ITTRAF	Number of times TRAFIC has been called in this assignment (for this $ASSIGN command).

c) Input files

Files are shown in the modules called by TRAFIC.

OUTPUT : Subroutine parameter

LOGICAL OK	TRUE unless set to FALSE by a module called by TRAFIC, indicating an error.

PROCEDURE : In the following description, names in the left hand column are subroutines called to perform the function described to the right.

1. CALCK	(variable demand only) Calculate component of demand function which is independent of automobile travel time.
2. NETWRK	Read in road network description into COMMON area /RNET1/.
3. BUSIN	(Bimodal assignment or user specified) Read bus volume on road links into COMMON area /RBUS/

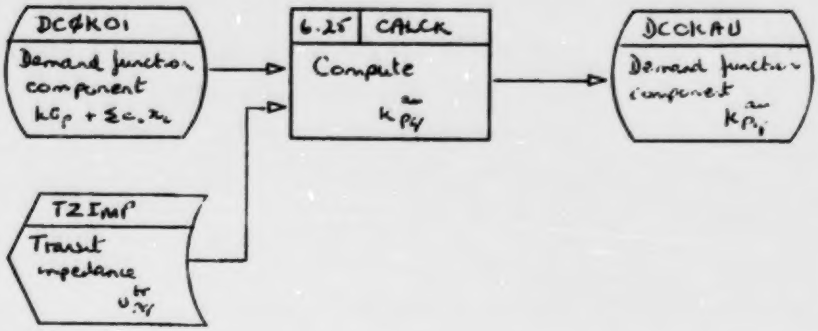
4. RASS Perform variable or fixed demand assignment giving predicted link times and volumes in COMMON area /RNET2/ and predicted demand (variable demand only) on file GPQRAN

5. RASSAV Save predicted link times and volumes on file RLTIVL, path times on file RZTIM and predicted demand (contents of GPQRAN) on file GPQAU.

6. Return.

MODULE: 6.25 CALCK

PURPOSE: Calculate the component of the demand function which is independent of the automobile travel time T^{au}_{pq} .


DCØKO1
Demand function component $kc_p + \Sigma c_i x_i$
6.25 CALCK
Compute k^{au}_{pq}
DCOKAU
Demand function component k^{au}_{pq}
TZIMP
Transit impedance u^{tr}_{pq}

INPUT: 1. File name: DC0K01
Type : Work File
Symbol : $k0_p + \sum c_i x_i = k01_{pq}$
Status : Required
Contents : Component of demand function which is independent of transit and automobile impedances. 1 record for each origin p containing the values of $k01_{pq}$ to all destinations q , q = 1, NCENT .
Sequence : Sequential by origin
Source : Created by 7.10 DC0S01

2. File name: TZIMP
Type : Intermediate
Symbol : u^{tr}_{pq}
Status : Required
Contents : Transit impedance. 1 record for each origin p containing the transit impedance from each origin p to all destinations q , q = 1 , NCENT
Sequence : Sequential by origin
Source : Created by 5.31 TIMPED or user specified

OUTPUT: File name: DC0KAU

Type : Work file

Symbol : K^{au}_{pq}

Status : Always created

Contents : Component of demand function which is independent of automobile travel time. One record for each origin p containing the demand function component K^{au}_{pq} to all destinations q , q = 1 , NCENT.

Sequence : Sequential by origin

Use : Input to 6.60 RASS

PROCEDURE: The program functions are as follows: -

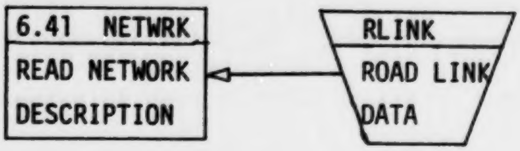
For each origin p , p = 1 , NCENT:

- Read demand function component $k01_{pq} = k0_{pq} + \sum c_i \; x_i$ for all destinations q , q = 1 , NCENT from file DC0K01
- Read transit impedance u^{tr}_{pq} for all destinations q , q = 1 , NCENT from file TZIMP .
- Calculate $K^{au}_{pq} = k01_{pq} + u^{tr}_{pq}$ for all q , q = 1 , NCENT
- Write demand function component K^{au}_{pq} , q = 1 , NCENT on file DC0KAU

Files DC0K01 and TZIMP are assumed to be sorted in ascending order by origin p .

Module : 6.41 NETWRK

Purpose : Read in link descriptions and construct tables



Input : File name : RLINK

Type : Data base

Symbol :

Status : Required

Contents : Data describing each link in the network

a) Essential data

- Link origin node or centroid
- Link destination node or centroid
- Distance (length)
- Link type

b) User defined data as read in subroutine LINKIN.
In the sample LINKIN this data is as follows

Sequence : Sequential in 2 sections

1. Access links from centroids
2. Links between nodes and egress links to centroids

Sections 1 and 2 are each sorted in ascending order by link origin and destination node. i.e. the links between nodes and the egress links are sorted together.

Source : User specified but separated into sections by the data base input program.

Procedure : The link descriptions are read in 2 steps:

1. Read essential data from RLINK and construct the following vectors. The link data is read in sequence for sections 1 and 2. The sequence number of the link in the input (L, L = 1, NLINK) continues from section 1 to section 2. File RLINK is rewound at the end of step 1 to allow step 2 to reread it if necessary.

Vector	Contents
NOLAB(NORIG)	Link number of first link L which has origin node or centroid number NORIG (NORIG = 1, NNODE where NNODE = maximum node number). Note 1. NOLAB(NORIG) = 0 if there is no link having origin node NORIG 2. NOLAB(NNODE + 1) is set to NLINK + 1 where NLINK = number of links in sections 1 and 2
ND(L)	Link destination node number of link L
DIST(L)	Length of link number L (miles)
LTYP(L)	Link type of link number (L)

2. Call user written subroutine LINKIN to read the link data necessary to supply the BPR curve parameters in subroutine FINDLC (called in module 6.60 RASS). A sample LINKIN will

read data into the following vectors. The sequence number of the links, L, L = 1, NLINK will correspond with the sequence number used for the essential data.

Vector	Contents
REAL NULA(L)	Number of lanes for link L

User written subroutine : LINKIN Read link characteristics and delay function data. This subroutine must read the link characteristics which are not part of the essential data (see RLINK) but which are required by subroutine FINDLC to determine the BPR curve for a given link. (see 6.60 RASS)

Subroutine statement:

```
SUBROUTINE LINKIN (NLINK)
```

where NLINK is the total number of links found when reading the essential data from RLINK.

Special assumptions : Module 6.60 RASS requires the following:

Centroid numbers p must satisfy $1 < p < NCENT$

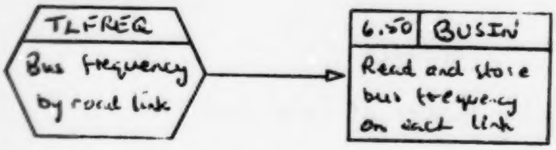
Node numbers i must satisfy $NCENT < i < NNODE$

where NCENT = maximum centroid number

NNODE = maximum node number

MODULE: 6.50 BUSIN

PURPOSE: Read and store bus frequency on each link.


TLFREQ
Bus frequency by road link
6.50
BUSIN
Read and store bus frequency on each link

INPUT: a) Data contained in COMMON areas.

The following vectors which are constructed by 6.41 NETWRK

NOLAB(NORIG)	Number, L, of first link with origin node NORIG
ND(L)	Link destination node of link L

b) Input Files

File name: TLFREQ
Type : Intermediate File
Symbol : f_a^{tr}
Status : Required
Contents : Number of vehicles per hour on each road link used by transit vehicles
Sequence : Sequential
Source : Output from 4.18 BUSFRQ.

OUTPUT: The following vector in a COMMON area:

BUSVOL(L)	Automobile equivalent of transit vehicles on link L.

PROCEDURE: For each link on file TLFREQ:

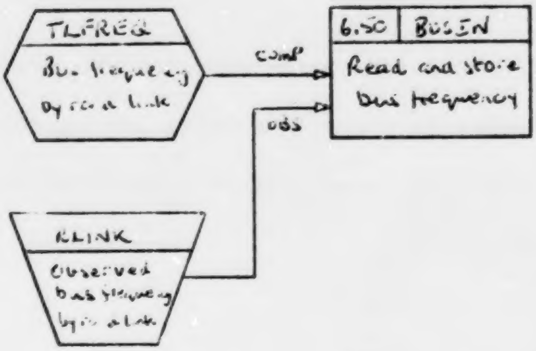
- Read in the link origin NORIG and destination NDEST nodes and the number of vehicles per hour.
- Search vectors NOLAB and then ND to find the link number L of the link with origin NORIG and destination NDEST
- Set BUSVOL(L) to fb_a

where $fb_a = f_a^{tr} * \alpha$

and α = automobile equivalent to 1 transit vehicle.

MODULE : 6.50 BUSIN

PURPOSE : Read and store bus frequency on each link.


TLFREQ
Bus frequency by road link
comp
obs
6.50
BUSIN
Read and store bus frequency
RLINK
Observed bus frequency by road link

INPUT : a) Data in COMMON areas.

The following vectors are constructed by 6.41 NETWRK

NOLAB(NORIG)	Number, L, of first link with origin node NORIG
ND(L)	Link destination node of link L.

b) Data in COMMON area /APARM/

INTEGER BUSFP	NO if no bus frequencies available
	COMP if bus frequencies are on file TLFREQ
	OBSx if bus frequencies are observed values in user data item x on RLINK.
REAL BUSFAC	Automobile equivalent of 1 transit vehicle (volume).

c) Input files

1. File name: TLFREQ

Type : Intermediate file

Symbol : f_a^{tr}

Status : Required for option COMP (see Procedure)

Contents : Number of transit vehicles per hour on each road link used by transit vehicles.

Sequence : Sequential

Source : Output from 4.18 BUSFRQ.

2. File name: RLINK

Type : Data base file

Status : Required for option OBS (see Procedure).

Contents : Number of transit vehicles per hour on each road road link, specified in a user defined data item.

Sequence : Sequential

Source : User defined

OUTPUT : The following vector in a COMMON area

REAL BUSVOL(L) Automobile equivalent of transit vehicles (buses) on link L.

PROCEDURE : 1. Check and set option defined by BUSFP. If BUSFP=NO, return.

2. (Option COMP)

For each link on file TLFREQ:

- read in link origin NORIG and destination NDEST nodes (internal numbers) and number of transit vehicles per hour.
- search vectors NOLAB and then ND to find the link number L of the link with origin NORIG and destination NDEST.
- set BUSVOL(L) to f_a^b

where $f_a^b = f_a^{tr} * \alpha$

and α = automobile equivalent of 1 transit vehicle (BUSFAC)

3. (Option OBS)

Determine number of user defined data item which contains transit vehicle frequency. BUSFP contains OBSx when x is the number of the data item.

Read each link record on RLINK using subroutine RLINKU (user defined data). The sequence number of the links L=1, NLINK will correspond to the sequence for the link data read in 6.41 NETWRK. For each link set BUSVOL(L) to f_a^b

where $f_a^b = \text{Obs}(x) * \alpha$

and Obs(x) is the observed transit vehicles per hour for the link, found in data item x

α is the automobile equivalent of 1 transit vehicle (BUSFAC).

4. Return.

MODULE : 6.60 RASS

PURPOSE: Perform the iterative traffic assignment procedure described in the paragraph PROCEDURE to predict the traffic volume and travel time on each link in the road network in order to satisfy the demand between each OD pair. The demand may be fixed or variable. A variable demand assignment requires additional data and results in a modified trip matrix to model a modal split compatible with the predicted travel time between each origin and destination. The variable demand assignment is used by the system's bimodal assignment procedure.
See File use diagram, Figure 1.

INPUT : a) Data contained in COMMON areas.

The following vectors are constructed by module 6.41 NETWRK

NOLAB(NORIG)	Number, L, of first link with origin node NORIG
ND(L)	Link destination node of link L
DIST(L)	Length of link L
LTYP(L)	Link type, link L. Turning links are identified by link types 90 thru 98.

The following vector is constructed by the sample user written input routine LINKIN for use with the sample user written routine FINDLC (see paragraph User written routine)

NULA(L)	Number of lanes on link L

The following vector is constructed by module 6.50 BUSIN

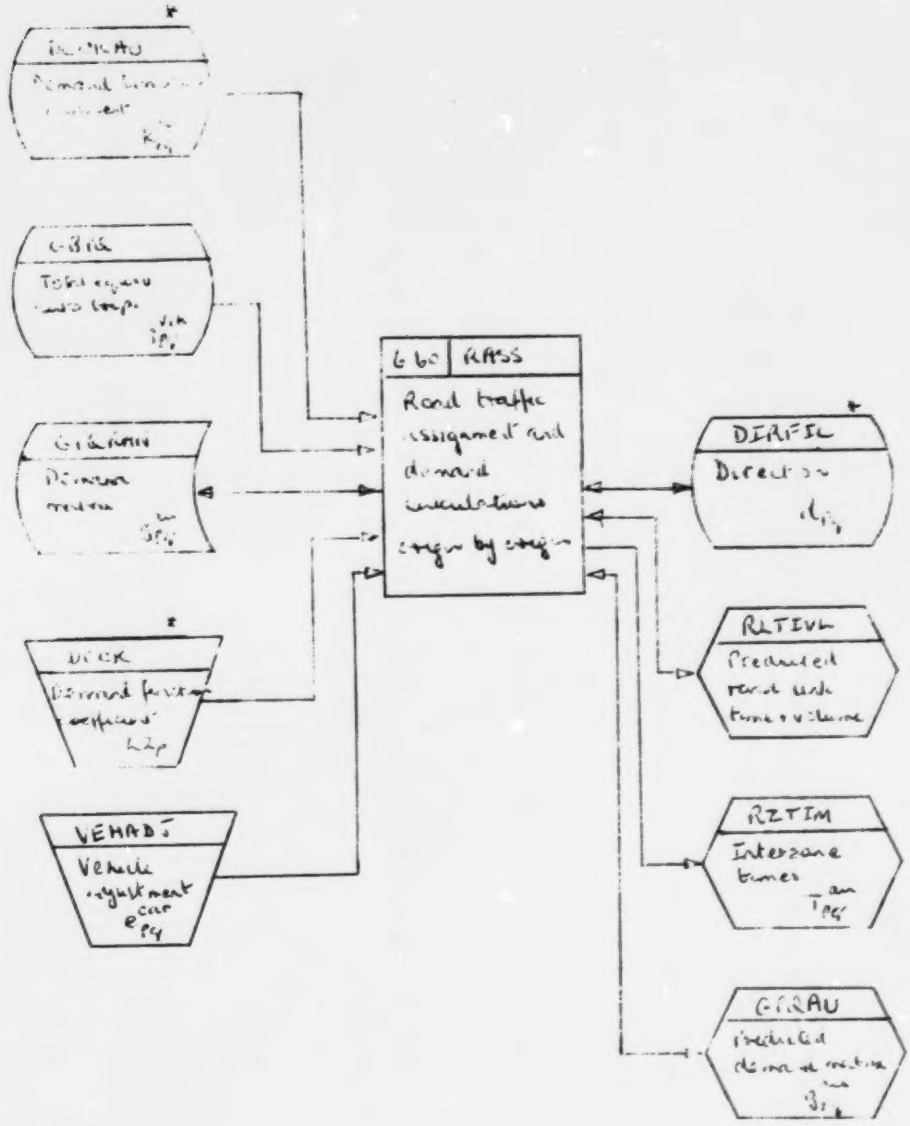
BUSVOL(L)	Automobile equivalent of transit vehicles on link L

Parameters used to control the assignment procedure in detail are contained in COMMON area /APARM/. These parameters are described in 1.60 DECASS and 1.62 DECPAR.

The user may define other vectors which are COMMON between LINKIN (see 6.41 NETWRK) and FINDLC (see paragraph "user written routine").

b) Input files:

See table of INPUT FILES. For details of file formats see "File Summary" section.


*
Total equiv. car trips
Demand matrix
*
Demand function coefficients
VEHADJ
Vehicle adjustment
e^{car}_{pq}
6.60
RASS
Road traffic assignment and demand calculations stage by stage
*
DIRFIL
Directions
d_{pq}
RLTIVL
Predicted road link time + volume
RZTIM
Interzone times
GRRAU
Predicted demand matrix

* Files required for Variable demand only

Figure 1 : RASS (and RASSAV) file use

File	Contents	Sequence	Source/Use	Status
INPUT FILES				
GBPQ	Vector of total trips $\bar{g}_{pq}^{veh}$ from origin p to destinations q, q = 1, NCENT	Sequential by origin p, p = 1, NCENT	Initialized by 6.10 AUTRIP	
GPQRAN	Vector of auto trips g_{pq}^{car} from origin p to destinations q, q = 1, NCENT	Random by origin p, p = 1, NCENT	Initialized by 6.10 AUTRIP or 6.15 INITDM	
DCOKAU	Vector of demand function components K_{pq}^{au} for origin p and destinations q, q = 1, NCENT	Sequential by origin p, p = 1, NCENT	Computed by 6.25 CALCK	Required for variable demand only
DCØK	Vector of demand function coefficients $k2_p$ for origin p	Sequential by origin p, p = 1, NCENT	Computed externally, data base file	Required for variable demand only
VEHADJ	Vector of values for vehicle adjustment from origin p to destination q, q = 1, NCENT	Sequential by origin p, p = 1, NCENT	Data base file	Required if vehicle adjustment used (ODEXSW = TRUE)
WORK FILES				
DIRFIL	Direction vectors (change in demand) from origin p to destinations q, q = 1, NCENT	Random by origin p, p = 1, NCENT	Updated each iteration	Required for variable demand only
GPQRAN	See INPUT FILE section		Updated each iteration	Updated for variable demand only

File	Contents	Sequence	Source/Use	Status
OUTPUT FILES	(Produced by 6.80 RASSAV)			
RLTIVL	Link origin and destination node numbers, predicted link volume and travel time	Sequential by link L L = 1, NLINK	1. Used to compute transit times 2. Used for reports	Always created.
GPQAU	Vector of predicted auto trips g_{pq}^{au} from origin p to destinations q, q = 1, NCENT	Sequential by origin p, p = 1, NCENT	1. For reports 2. Compute g_{pq}^{tr} in ODTRAN	Always created.
RZTIM	Vector of interzone travel times by shortest path T_{pq}^{au} from origin p to destinations q, q = 1, NCENT	Sequential by origin p, p = 1, NCENT	1. For reports 2. Compute K_{pq}^{tr} in 5.30 CALKTR	Always created

USER WRITTEN
ROUTINE: FINDLC Retrieve link characteristics

This subroutine is called from the functions USECST and TLCST which are part of the assignment procedure. The subroutine must determine the link characteristics required to calculate the travel time on a given link in the road network using the BPR delay curve. These characteristics are: the free flow speed, the practical capacity and BPR curve coefficients a and b . FINDLC may call other routines written by the user.

Subroutine statement:

```
SUBROUTINE FINDLC (L,LTYPE, SPDFF, CAP, RNULA, A, B)
```

L, LTYPE are input data to FINDLC

INTEGER L = link sequence number as described for module 6.41 NETWRK

INTEGER LTYPE = link type of the link

SPDFF, CAP, RNULA, A, B must be determined in FINDLC or the subroutines it calls.

REAL SPDFF = free flow speed on the link L

REAL CAP = practical capacity of link L

REAL RNULA = number of lanes on link L. (This parameter is not used in RASS but may be required by other calls for link characteristics)

REAL A = coefficient a of BPR curve

REAL B = coefficient b of BPR curve

COMMON statements:

FINDLC will probably contain the same COMMON statements as LINKIN (See module 6.41 NETWRK) which reads or defines the user data necessary to determine the link characteristics.

The sample FINDLC refers to the COMMON/USER/ which contains the number of lanes on each link, read in by LINKIN. The sample FINDLC calls FINDCC to determine the characteristics which depend on the link type only. (Curve characteristics). Thus FINDCC contains the characteristics in vectors initialized in FINDCC itself.

PROCEDURE : In the following description, sections that apply to variable demand only are so indicated. The name in the left hand column are subroutines or functions which contain code for the section described to the right.

A. Adjust initial demand (variable demand only)

1. (Variable demand only) Check initial demand for all OD pairs, file GPQRAN, and adjust to within upper bound of demand function (value of function with zero auto impedance).

B. Find initial solution

2. If this is not first traffic assignment (bimodal assignments), read in previous solution from file RLTIVL. Go to step 4.

3. Determine all or nothing solution:

USECST 3.1 For all links a,a=1,NLINK: compute link travel time for zero link volume

3.2 Assign link volume for all or nothing solution i.e. using times from step 3.1.
For all origins p , p=1, NCENT
- Read OD matrix g_{pq}^{car}, all destinations q from file GPQRAN

(LOADTR) MINTRE - Trace shortest paths using times from step 3.1

LOADTR - For all destinations q, q=1, NCENT, accumulate trips onto links on path pq (= all or nothing volume)

USECST 3.3 For all links a, a=1, NLINK: compute link travel time using volume obtained in step 3.2

4. OBJFN compute objective function

TLCST 4.1 Sum for all links a, a=1, NLINK: Fixed demand component of objective function using volume obtained in step 3.

4.2 (Variable demand only) For all origins p, p=1, NCENT
- Read OD matrix g_{pq}^{car}, all destinations q from file GPQRAN
- Read total trip matrix $\bar{g}_{pq}^{veh}$, all destinations q from file GBPQ

- Read demand function component K_{pq}^{au}, all destinations q, from file DCOKAU and coefficient $k2_p$ from file DCOK

TDCST	- Accumulate for all destinations q, q=1, NCENT, variable demand component of objective function.

C. Adjust link volumes and (for variable demand only) OD demand g^{au}_{pq} untill convergence criteria are satisfied.

Steps 5 thru 11 are repeated for each iteration,

5. Assign volume to links using current time; adjust demand for variable demand only.

   For all origins p, p=1, NCENT

   - Read OD matrix (demand g^{car}_{pq} to all destinations q, q=1, NCENT from file GPQRAN
   - Read vehicle adjustment matrix e^{car}_{pq} to all destinations q. q=1, NCENT from file VEHADJ (if VEAD=yes)
   - (Variable demand only) Calculate implied impedance u^{au}_{pq}
     . Read total trip matrix $\bar{g}^{veh}_{pq}$ all destinations q from file GBPQ
     . Read demand function component K^{au}_{pq}, all destinations q, from file DCOKAU and coefficient $k2_p$ from file DCOK

DEMCST . Calculate implied impedance u^{au}_{pq}, all destinations q

(LOADTR) MINTRE - Determine shortest paths from p to q, q=1, NCENT, using current time c_a

LOADTR - (Variable demand only) Adjust demand g^{au}_{pq} if travel time $T^{au}_{pq} \neq$ implied impedance u^{au}_{pq}

LOADTR - Accumulate trips g^{au}_{pq} to each destination q onto links on shortest paths (= final volume)

- Write direction vector d_{pq} (= adjustment to demand) on DIRFIL

6. Find λ to minimize objective function.

   6.1 For all links a, a=1, NLINK: compute difference between volume and current volume. (= volume difference)

6.2 For each possible value of λ compute objective function and determine value of λ giving objective function value (λ*).

TLCST - Sum for all links a, a = 1, NLINK: Fixed demand component of objective function, calculated using: -
volume = previous volume + (λ *volume difference)

- (Variable demand only). For all origins p, p = 1, NCENT:
  - . Read direction vector (adjustments to demand for each destination) from DIRFIL
  - . Read current demand from GPQRAN
  - . Read demand function component K_{pq}^{au}, all destinations q from file DCOKAU and coefficient $k2_p$ from file DCOK

TDCST - Accumulate variable demand component of objective function using: -
demand = previous demand + λ * adjustment to demand

- Determine value of λ^* giving minimum objective function

7. Update link volume using λ^* determined in step 6.2
For all links a, a = 1, NLINK: Set
volume = previous volume + (λ^* * volume difference)

8. USECST For all links a, a = 1, NLINK. Compute link travel time c_a using volume calculated in step 7.

9. (Variable demand only) Adjust demand.

For all origins p, p = 1, NCENT

- Read direction vector d_{pq} (= adjustment to demand) from DIRFIL
- Read previous demand vector from GPQRAN

- Compute, for all destinations q, q = 1, NCENT

  $$\text{demand}_{pq} = \text{previous demand}_{pq} + (\lambda^{*} * d_{pq})$$

- Write new demand vectors in place of previous demand vectors on file GPQRAN

10. Test change in objective function

    Calculate lower bound of objective function. Test difference between objective function and lower bound. If less than required percentage go to step 12, otherwise go to step 11.

11. Test other terminating criteria.

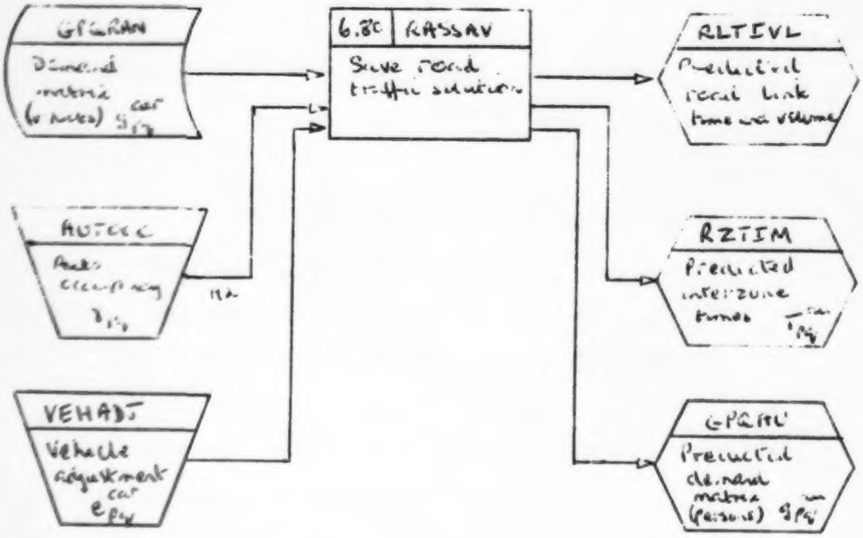
    Test if number of iterations or running time exceed user specified limits. If so go to step 12 otherwise continue assignment with another iteration i.e. go to step 5.

12. MOES Calculate total vehicle hours and miles

13. Return.

MODULE : 6.80 RASSAV

PURPOSE : Save road traffic assignment solution values.


GPQRAN
Demand matrix (vehicles) g^{car}_{pq}
6.80
RASSAV
Save road traffic solution
RLTIVL
Predicted road link time and volume
AUTOCC
Auto occupancy λ_{pq}
RZTIM
Predicted interzone times $\bar{t}^{car}_{pq}$
VEHADJ
Vehicle adjustment e^{car}_{pq}
GPQAU
Predicted demand matrix (persons) g^{car}_{pq}

INPUT : a) Data in COMMON area /NETSIZ/

NCENT	Highest centroid number (internal)
NNODE	Highest road node number (internal)

b) Data in COMMON area /UNIT/

INTEGER RLTIVL	Unit number of road traffic time and volume file.
INTEGER RZTIM	Unit number of auto interzone time matrix file.
INTEGER GPQAU	Unit number of predicted auto demand matrix file.

c) Data in COMMON area /APARM/

LOGICAL BVOLSW	TRUE indicates bus frequency was used in assignment (parameter BUSF=COMP or OBSx). FALSE otherwise.
INTEGER ITRD	Number of iterations executed in traffic assignment.
INTEGER IDEN(10)	Identification text from subcommand IDENT
LOGICAL RVARSW	TRUE if assignment was with variable demand. FALSE otherwise.
LOGICAL ODEXSW	TRUE if vehicle adjustment matrix was used, FALSE otherwise

INTEGER AOCC REAL AOCCV	Auto occupancy parameter (see description in 1.62 DECPAR). Options in the PROCEDURE are determined as follows: If AOCC=DEFA or VALU, option is A1 and γ = default or AOCCV If AOCC=PERS, option is A2 and γ_{pq} will be read from AUTOCC.

d) Data in COMMON areas /RNET1/,/RNET2/,/RBUS/; N=1,NNODE; L=1,NLINK

INTEGER NOLAB(N)	Link number of first link with origin node N (internal number).
INTEGER ND(L)	Destination node number of link L.
REAL VCUM(L)	Predicted volume on link L.
REAL TRAV(L)	Predicted travel time on link L.
REAL BUSVOL(L)	If bus frequency was used in assignment, contains auto equivalent of buses on link L.

e) Input file:

1. File name: GPQRAN (#602)
   Type : Work file
   Symbol : g_{pq}^{car}
   Status : Required
   Contents : Matrix of auto trips between each origin and all destinations.
   Sequence : Random, indexed by origin.
   Source : Created by 6.60 RASS.

2. File name: AUTOCC

   Type : Data base

   Symbol : γ_{pq}

   Status : Required for option A2 only
   (See PROCEDURE)

   Contents : Auto occupancy between origin and destination centroid

   Source : User specified

   Sequence : Read in sequence by origin

3. File name: VEHADJ

   Type : Data base

   Symbol : e^{car}_{pq}

   Status : Required if variable ODEXSW in COMMON/APARM/ is TRUE
   (Subcommand parameter VEAD=YES)

   Contents : Vehicle adjustment matrix as number of vehicles to be added to those required for person trips between each origin and each destination

   Sequence : Sequential by origin

   Source : User specified

OUTPUT : a) Output files

1. File name: RLTIVL (#631)

   Type : Intermediate file

   Status : Always created

   Contents : Link origin and destination node numbers, predicted auto volume and travel time for each link in road network.

   Use : Input to 6.94 PRTLNK via 6.95 RDRLTV. (Link report).
   Input to transit time calculation

2. File name: RZTIM (#632)
   Type : Intermediate file
   Status : Always created
   Contents : Predicted interzone auto travel times
   Use : Input to 5.30 CALKTR
   For reports.

3. File name: GPQAU (#633)
   Type : Intermediate file
   Status : Always created
   Contents : Predicted demand matrix of person trips by automobile
   Use : To calculate transit demand in bimodal assignment (5.41 ODTRAN).
   For reports.

b) Subroutine parameter:

LOGICAL OK FALSE if files AUTOCC and GPQRAN are used and do not match. TRUE otherwise.

PROCEDURE: RASSAV calculates the predicted number of person trips by automobile using the same options for the auto occupancy as were used in the initializing module 6.10 AUTRIP that is:

Option A1: $g^{au}_{pq} = (g^{car}_{pq} - e^{car}_{pq})* \ \gamma$

Option A2: $g^{au}_{pq} = (g^{car}_{pq} - e^{car}_{pq})* \ \gamma_{pq}$ q = all destinations

See module 6.10 AUTRIP for details of the 2 options.

e^{car}_{pq}, the vehicle adjustment, is assumed to be zero if ODEXSW in /APARM/ is FALSE (subcommand parameter VEAD = NO)

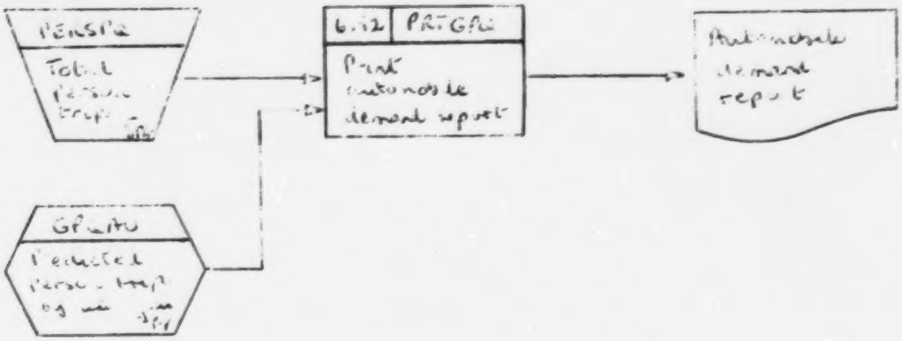
In the following description, names in the left hand colum are subroutines which contain code for the section described to the right.

1. Write identification record on file RLTIVL
2. For each origin node
   - convert node number to external
   - For each link with this node as origin
     . convert destination node number to external

```
                .write external origin and destination node numbers,
                  predicted auto volume, travel time and bus volume to
                  file RLTIVL (Bus volume zero if none used in
                  assignment).
3.              Write end file and rewind file RLTIVL.
4.              Write identification on file RZTIM.
5.              For each origin centroid:
     MINTRE     - trace shortest path to all destinations using
                  predicted travel times.
                - write shortest path times to all destinations on
                  file RZTIM.
6.              Write end file and rewind file RZTIM.
7.              Write identification record on file GPQAU
8.              Determine auto occupancy option from AOCC
                A1 Global value for auto occupancy
                A2 Auto occupancy specified for each OD pair.
9.              (Option A2)  Rewind AUTOCC
10.             For each origin centroid:
                - Read predicted demand from file GPQRAN
                - (If ODEXSW = TRUE)  Read vehicle adjustment from file VEHADJ.
                - (Option A2) read auto occupancy from file AUTOCC.
                - If files do not match, print a message, set OK to
                  FALSE and return.
                - For each destination
                  . calculate person trips by automobile according to
                    the option (A1 or A2)
                - Write person trips by automobile for all destinations
                  to file GPQAU
11.             Write end file and rewind GPQAU
12.             Return.
```

MODULE : 6.92 PRTGPQ

PURPOSE : Print predicted automobile demand report.


PERSPQ
Total person trips
6.92
PRTGPQ
Print automobile demand report
Automobile demand report
GPQAU
Predicted person trips

INPUT : a) Data in COMMON area /NETSIZ/

NCENT Highest centroid number (internal)

b) Data in COMMON area /UNIT/

INTEGER GPQAU Unit number of file GPQAU

c) Input files:

1. File name: PERSPQ

Type : Data base

Symbol : $\bar{g}_{pq}$

Status : Required

Contents : OD matrix of number of person trips between each origin and each destination centroid.

Sequence : Read in sequence by origin centroid.

2. File name: GPQAU

Type : Intermediate file

Symbol : g_{pq}^{au}

Status : Required

Contents : Predicted demand by auto, i.e. number of person trips between each origin and each destination predicted by a road traffic assignment with variable demand, or bimodal assignment.

Source : Output from 6.80 RASSAV.

Sequence : Read in sequence by origin centroid.

OUTPUT : a) Printed report

The report gives for each row (origin) in the OD matrix

- number of predicted person trips by auto to all destinations (Predicted demand)
- the number of predicted person trips by auto as a percentage of the total trips, to all destinations.

The report is printed in the order of rows on the matrix files. For each matrix row,data is printed with 10 destinations per line.

b) Subroutine parameter

LOGICAL OK TRUE if no catastrophic errors are found;

FALSE if one of the following errors is found:

- an external node number greater than 9999 is encountered.
- the OD matrix file of total trips in the data bank contains less records (rows) than the file of predicted person trips by auto (GPQAU).

PROCEDURE : File 221 PERSPQ is accessed by routines specified in its file description.

1. Find maximum external node number of centroids
2. Read and print identification record from file GPQAU.

3. Print standard report title (subroutine TTIT).
4. For each OD matrix row
   - Read predicted auto demand record from file GPQAU. If end of file is encountered, rewind GPQAU and return.
   - Read total trips from file PERSPQ. If end is encountered, print an error message, set OK=FALSE and return.
   - Initialize index vector to 1 plus maximum(= number of centroids + 1). The index vector is as long as the maximum external centroid number.
   - For each destination with non-zero total trips
     . convert destination centroid number to external number.
     . if predicted demand is zero, add a zero value to predicted demand record and set percentage to -1.0.
     . if predicted demand is non zero, calculate percentage of total trips.
     . set element of index vector that corresponds to the destination external node number, to position of predicted demand in predicted demand record.
   - Scan index vector in groups 1-9, 10-19, 20-29, etc. (external destination node number). If any group contains a non zero value for total trips, indicated by a position number in the predicted demand record, print the data for that group.

MODULE : 6.94 PRTLNK

PURPOSE : Print road traffic assignment link report.

INPUT : (Depends on setting of subroutine parameter DATAIN)

a) Subroutine parameter:

LOGICAL DATAIN	TRUE indicates that the road network data is in COMMON areas /RNET1/, /RNET2/ and /RBUS/. FALSE indicates that data must be read into those COMMON areas from the data bank and file RLTIVL by subroutines 6.41 NETWRK and 6.95 RDRLTV.
LOGICAL CORSW	TRUE indicates that corridor specifications should be read and only links in corridor included in the report.

b) Data in COMMON areas /RNET1/, /RNET2/, /RBUS/ (only applicable if DATAIN=TRUE) where L=1,NLINK and N=1,NNODE

REAL DIST(L)	Length of link L
INTEGER LTYP(L)	Link type of link L
INTEGER NOLAB(N)	Link number of first link with origin node N (internal number)
INTEGER ND(L)	Destination node of link L (internal number)
REAL VCUM(L)	Predicted autos per hour on link L
REAL TRAV(L)	Predicted auto travel time on link L (minutes)
REAL BUSVOL(L)	Auto equivalent of transit vehicles on link L

c) Data in COMMON area /NETSIZ)

NCENT	Highest centroid number (internal)
NNODE	Highest (road) node number (internal)

d) Data in COMMON area /APARM/

REAL BUSFAC	Number of autos equivalent to 1 transit vehicle (for link volume)

e) Input file

File name : RLINK (#211)

Type : Data base

Status : Required

Contents : observed values for each link in the network

- observed travel time (-1. if not specified)
- observed volume (-1. if not specified).

Sequence : Sequential. Sorted in ascending order by link origin and destination nodes (internal numbers).

Source : User specified but sorted by the data base input program.

OUTPUT : a) Printed report

1 line for each link, sorted into ascending order by link origin and destination nodes (external numbers).

For each link the data printed are:

- link sequence in report
- link origin node number (external number)
- link destination node number (external number)
- link length (miles)
- link type
- free flow speed (determined from link type, m.p.h.)
- capacity (determined from link type, veh per hour).
- observed travel time (mins)
- observed volume (veh per hour)
- predicted speed (m.p.h.)
- predicted travel time (mins)
- predicted volume (veh per hour)
- number of lanes
- number of buses (buses per hour).

b) Subroutine parameter

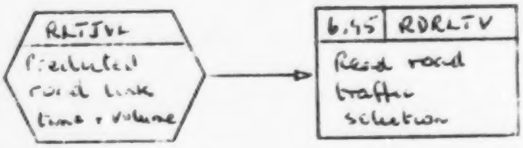
LOGICAL OK TRUE if input successful
FALSE if errors in subroutine NETWRK or RDRLTV.

PROCEDURE : In the following description, names in the left hand column are subroutines which contain code for the section described to the right.

1.		If DATA IN is false,
	NETWRK	- Read data into COMMON/RNET1/ and user data required for delay function computations (see 6.41 NETWRK)
	RDRLTV	- Read in values, predicted by road traffic assignment, from file RLTIVL (see 6.95 RDRLTV)
	CORRD2	Read in corridor specifications as a list of nodes. Construct vector LFLAG, such that LFLAG(L) is TRUE if both the origin and destination nodes of link L are in the corridor
	MOES	Calculate and print vehicle hours and miles
2.		Read in observed data from file RLINK
3.	GTNODV	Get external node numbers for all nodes
	QSORT	Sort external numbers (and an associated vector of internal node numbers) into ascending order.
4.		For each number in sorted external node number list - pick up first link number of link with this node as origin - determine last link with this node as origin - get internal node number of destination nodes of these links - order these links by external destination number, including only those with corridor, i.e. with LFLAG(L) = TRUE, if CORSW is true. - print 1 line for each of these links in this order.
5.		Return.

MODULE : 6.95 RDRLTV

PURPOSE : Read in road traffic assignment solution.



INPUT : a) Data in COMMON area /RNET1/

INTEGER NOLAB(N) Link number of first link with origin node N (internal number), N=1, NNODE

INTEGER ND(L) Destination node number of link L, L=1 NLINK

b) Data in COMMON area /NETSIZ/

NNODE Highest (road) node number (internal)

NLINK Number of road links

c) Data in COMMON area /UNIT/

INTEGER RLTIVL Unit number of file RLTIVL

d) Input file:

File name: RLTIVL (#631)

Type : Intermediate file

Status : Required

Contents : Predicted auto volume and travel time for each link in the road network

Source : Output from 6.80 RASSAV

OUTPUT : a) Subroutine parameter

LOGICAL OK TRUE if no catastrophic errors are found.

FALSE if solution is unusable (See Procedure).

b) Data in COMMON area /RNET2/ and /RBUS/; L=1, NLINK

REAL VCUM(L)	Predicted autos per hour on link L
REAL TRAV(L)	Predicted auto travel time on link L
REAL BUSVOL(L)	Auto equivalent of transit vehicles on link L.

PROCEDURE : 1. Read identification record from file RLTIVL and print identification.

2. Initialize VCUM, TRAV and BUSVOL to zero and IFOUND to FALSE for all links

3. Read a link record from RLTIVL. If end of file is reached, go to step 7.

4. Convert origin and destination node numbers to internal numbers.

5. Check this link is in network (vectors NOLAB and ND). If it is not and the predicted volume is non zero, print a message and set OK to FALSE, them go to step 3.

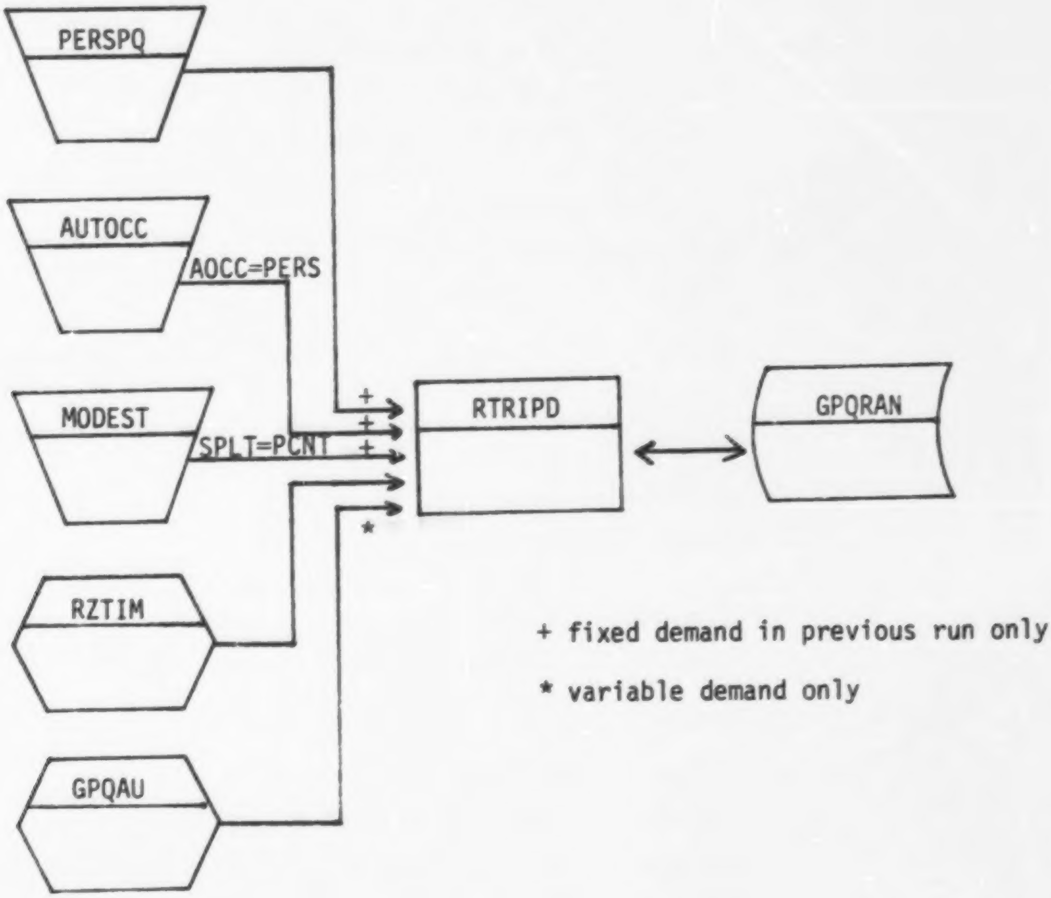
6. (link found in network) Place solution values for volumes and travel time in corresponding link position in vectors VCUM, BUSVOL and TRAV.

7. (EOF on RLTIVL) Check that, for all links in the network, a record was found on the solution file. Print a message for those links for which no record was found (Volumes and time remain zero).

8. Rewind RLTIVL and return.

MODULE : 6.96 RTRIPD

PURPOSE : Print road network trip time distribution


PERSPQ
AUTOCC
AOCC=PERS
MODEST
SPLT=PCNT
RZTIM
GPQAU
RTRIPD
GPQRAN
+
+
+
*
+ fixed demand in previous run only
* variable demand only

INPUT : a) Input files

1. File name : PERSPQ
   Type : Data bank
   Symbol : $\bar{g}_{pq}$
   Status : Required only if fixed demand assignment in a previous execution of EMME
   Contents : OD matrix of number of person trips between each origin and each destination centroid
   Source : User specified
   Sequence : Read in sequence by origin

2. File name : AUTOCC
Type : Data bank
Symbol : γ_{pq}
Status : Required only if fixed demand assignment in a previous execution of EMME and AOCC=PERS
Contents : Auto occupancy between each origin and each destination centroid
Source : User specified
Sequence : Read in sequence by origin

3. File name : MODEST
Type : Data bank
Symbol : m_{pq}
Status : Required only if fixed demand assignment in a previous execution of EMME and SPLT=PCNT or PERS
Contents : Modal split estimate as percentage or number of trips
Sequence : Sequential by origin
Source : User specified

4. File name : RZTIM
Type : Intermediate file
Symbol : T^{au}_{pq}
Status : Required
Contents : Predicted interzone auto travel times
Sequence : Sequential by origin
Source : Created by 6.80 RASSAV

5. File name : GPQAU
Type : Intermediate file
Symbol : g^{au}_{pq}
Status : Required if assignment was variable demand
Contents : Predicted demand matrix of person trips by automobile
Sequence : Sequential by origin
Source : Created by 6.80 RASSAV

b) Subroutine parameters

LOGICAL VEHFIL TRUE if file GPRRAN has already been created in this execution of EMME.
FALSE otherwise.

OUTPUT : Printed report.
The report gives for a series of intervals of interzone travel time, the number of person trips which take a time within the interval.

PROCEDURE : In the following description, names in the left hand column are subroutines which contain code for the section described to the right.

1. If assignment was with variable demand go to step 4

2. If GPQRAN exists (VEHFIL=TRUE) go to step 4

3. RINIT Compute vehicle equivalent of auto demand on file GPQRAN

4. Decode auto occupancy option

5. For each origin p, p=1,NCENT

- Read interzone times T_{pq}^{au} from file RZTIM
- (Fixed demand) Read vehicle equivalent of auto demand g_{pq}^{car} from file GPQRAN. If AOCC=PERS, read auto occupancy from file AUTOCC. Compute person trips by auto

$$g_{pq}^{au} = g_{pq}^{car} * \gamma_{pq}$$

UPACOD - (Variable demand) Read person trips by auto g_{pq}^{au} from file GPQAU

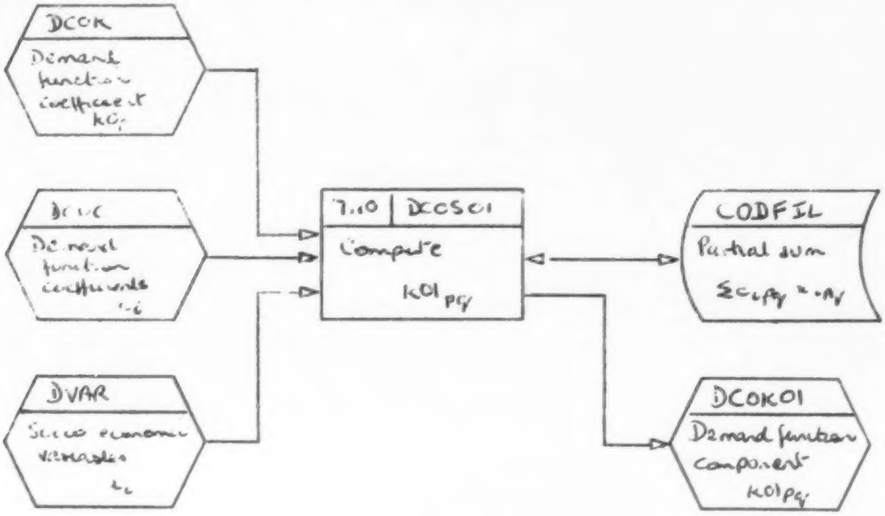
- For each destination q, q=1,NCENT
  . Determine interval (i) for interzone time T_{pq}^{au}
  . Add number of trips g_{pq}^{au} to accumulator for interval i

6. Print frequency report

7. Return.

MODULE: 7.10 DC0S01

PURPOSE: Calculate the component of the demand function which is independent of automobile and transit impedances.


DCOK
Demand function coefficient k0
DCUC
Demand function coefficients ci
DVAR
Socio economic variables xi
7.10 DCOS01
Compute k01pq
CODFIL
Partial sum Σcipq xipq
DCOK01
Demand function component k01pq

INPUT: 1. File name: DC0K

Type : Data base

Symbol : $k0_p$, $k2_p$, $k3_p$

Status : Required

Contents : Coefficients of demand function. 1 record for each origin p containing the value of the coefficients k0 , k2 , k3 for that origin

Sequence : Sequential by origin

Source : User specified, result of demand function calibration

2. File name: DC0C
Type : Intermediate file
Symbol : c_i or c_{ip} , c_{ipq}
Status : Required
Contents : Coefficients applicable to the socio-economic variables (x_i) in the demand function. For coefficients which vary by origin zone p (type A), 1 record for each coefficient containing the coefficient number, an origin indicator and the values of the coefficient for each zone c_{ip} , $p = 1$, NCENT

For coefficients which vary by origin zone and destination zone (type B), 1 record for each origin p , $p = 1$, NCENT for each coefficient containing the coefficient number and the values of the coefficient between the origin and all destinations q, $q = 1$, NCENT (See also the file description)

Sequence : 1 record for each coefficient of type A followed by NCENT records for each coefficient of type B. The records for type B coefficients are grouped by coefficient in ascending order of origin p, $p = 1$, NCENT.
Source : User specified, result of demand function calibration

3. File name: DVAR
Type : Data base file
Symbol : x_i or x_{ip} , x_{iq} , x_{ipq}
Status : Required
Contents : Values of the socio-economic variables with demand function. The file format is similar to that of DC0C, except that variable of type A may vary by destination with 1 record for each destination q containing x_{iq}, $q = 1$, NCENT
Sequence : Similar to DC0C
Source : User specified

OUTPUT: File name: DC0K01
Type : Intermediate file
Symbol : $k01_{pq}$
Status : Always produced
Contents : Component of demand function which is independent of automobile and transit impedances. 1 record for each origin p containing the values of $k01_{pq}$ to all destinations q , q = 1 , NCENT
Sequence : Sequential by origin p , p = 1 , NCENT
Use : Input to 6.25 CALCK

WORK FILE: File name: CODFIL
Type : Work file
Symbol : $\sum_{ipq} c_{ipq} \; x_{ipq}$
Status : Always required
Contents : Partial sum of components of $\sum_i c_i x_i$ which vary by origin and destination zones. 1 record for each origin p containing the values of $\sum_i c_{ipq} \; x_{ipq}$ for all destinations q , q = 1 , NCENT .
Sequence : Random, indexed by origin p
Use : To accumulate the partial sum $\sum_i c_{ipq} \; x_{ipq}$.

PROCEDURE: The program accumulates the following sum

$$k01_{pq} = k0_p + \sum_i c_{ip} x_{ip} + \sum_i c_{ip} x_{iq} + \sum_i c_{ipq} x_{ipq}$$

The components of the sum are accumulated in the following vectors

Vectors	Contents
CXORIG (IP)	$kO_p + \sum_i c_{ip} x_{ip}$, p = 1 , NCENT
CXDEST (IQ)	$\sum_i c_{ip} x_{iq}$, q = 1 , NCENT
CXOD (IQ)	$\sum_i c_{ipq} x_{ipq}$, q = 1 , NCENT

where	ip	indicates variables specified for the origin zone p
	iq	indicates variables specified for the destination zone q
	ipq	indicates variables specified for the trip from origin p to destination q .
	CXOD	is reused for each origin p .

The program functions are as follows: -

1. For each origin p , p = 1 , NCENT
    - Read demand function coefficients (k) from file DC0K
    - Set CXORIG (IP) = kO_p
2. For each destination q , q = 1 , NCENT
    - Set CXDEST (IQ) = 0
3. Initialize file CODFIL to zeros.
4. - Read a record of demand function coefficients c_i from file DC0C (includes coefficient number and OD indicator or origin number) if first record or previous coefficient processed.
    - Read a record of values of a socio-economic variable, x_i , from file DVAR (includes variable number) if first record or previous variable processed.

- Check that the coefficient and variable numbers are same

  If coefficient number is lower (coefficient only) signal an error and go to step 4 to read another coefficient record only.

  If coefficient number is higher (variable only) go to step 4 to read another variable record only.

- If EOF is found go to step 5.

- If the OD indicator is 'origin' for both coefficient and variable, for each origin IP = p, p = 1, NCENT compute

  $$CXORIG(IP) = CXORIG(IP) + C_{ip}\ x_{ip}$$

  Go to step 4 to read coefficient and variable records.

- If the coefficient OD indicator is 'origin' and the variable OD indicator is 'destination', for each origin IP = p, p = 1, NCENT
  - . Read the record for origin p into CXOD from file CODFIL
  - . Compute for each destination IQ = q, q = 1, NCENT

    $$CXOD(IQ) = CXOD(IQ) + c_{ip}\ x_{iq}$$

  - . Rewrite CXOD to file CODFIL

  Go to step 4 to read coefficient and variable records.

- If the OD indicator specifies an origin number p
  - . Read the record for origin p into CXOD from file CODFIL
  - . Compute for each destination IQ = q, q = 1, NCENT

    $$CXOD(IQ) = CXOD(IQ) = c_{ipq}\ x_{ipq}$$

  - . Rewrite CXOD to file CODFIL
  - . Go to step 4 to read coefficient and origin records

5. (EOF on coefficient and variable file) For each origin IP = p, p = 1, NCENT
   - Read the record for origin p into CXOD from file CODFIL
   - For each destination IQ = q, q = 1, NCENT, compute

     $$CXOD(IQ) = CXOD(IQ) + CXORIG(IP) + CXDEST(IQ)$$

   - Write vector CXOD on the file DC0KO1

Note: This procedure supposes that files DC0C and DVAR are sorted into similar order by coefficient (and by origin). The routine will also handle the case of coefficient and variable by destinations.That is CXDEST(IQ) = CXDEST(IQ) + $C_{iq}\ x_{iq}$

VARIABLES : Possible variables are the following:

a) Applicable to the origin

Population
Car ownership
Income
Household size

b) Applicable to the destination

Employment
Parking availability
Auto parking cost

c) Varying by origin and destination

Transit fare
Auto milage cost

MODULE : 8.01 PTITLE

PURPOSE : Plot title and date on vehicle and transit plots.

INPUT : a) Data in COMMON block /GEOM/

WSYM Quotient width/height for a plotted character using the system supplied plot routine SYMBOL*.

b) Subroutine parameters

XU Largest x-coordinate value for nodes in plotted region

YU Largest y-coordinate value for nodes in plotted region.

c) Input file

1. File name : PTITDT
Type : Direct entry input file
Symbol :
Status : Required
Contents : Title for plot
Sequence : Sequential
Source : User specified

OUTPUT : Subroutine parameters

XV Amount by which the x-coordinate of each node must be displaced to ensure that the plot title is centred on plot.

YTIT y-coordinate of title

PROCEDURE : 1. Read in title from file PTITDT

2. Determine date (routine uses CDC system routine TDAT)

3. Calculate x and y coordinates for the title. If the title length is less than the x-range of the network, centre title on network. Otherwise determine XV, the amount by which the x-coordinate of each node must be displaced to ensure the network is centred on

---

* CDC system plot routine

the plot with respect to the title.

4. Plot title and date

5. Return

SYSTEM PLOT
ROUTINES : SYMBOL

MODULE : 8.02 HEADNG

PURPOSE : Plot distance scale and direction indicator on vehicle and transit plots.

INPUT : a) Data in COMMON block /GEOM/

SCALE	Scale of plotted network (ft./in.)
ANGROT	Angle in counter clockwise direction between R and R' where R is the system of reference of the geo-coded coordinates and R' is the system of reference of the plot.
ANGNOR	Angle between x-axis and north in counterclockwise sense in system of reference R.
WSYM	Quotient width/height for a plotted character using the system supplied plot routine SYMBOL

b) Subroutine parameters

XU	Largest x-coordinate value for nodes in plotted region.
YU	Largest y-coordinate value for nodes in plotted region.
XV	Amount by which the x-coordinate for each node must be displaced to ensure that the plot title is centred on plot.
YTIT	y-coordinate of title.

PROCEDURE :
1. Plot north direction indicator
2. Plot scale
3. Return

SYSTEM PLOT ROUTINES : PLOT, NUMBER, SYMBOL. *

* CDC system plot routines.

CF 24/4/78

MODULE : 8.03 COMPL

PURPOSE : Logical function checking if all links exiting a given node have been plotted.

INPUT : Function parameter
NEX node number

OUTPUT : COMPL TRUE if all links exiting node NEX have been plotted
Otherwise FALSE.

PROCEDURE : 1. Set COMPL to .FALSE.
2. Calculate number of links exiting NEX which have been plotted.
If this equals the total number of exiting links set COMPL to .TRUE.
3. Return.

CF 24/4/78

MODULE : 8.04 ANGLE

PURPOSE : Function finds angle of inclination of a given line to x-axis.

INPUT : Function parameters

X1, Y1 x and y coordinates of beginning of line

X2, Y2 x and y coordinates of end of line.

OUTPUT : ANGLE angle of inclination in radians of line to x-axis

$(0 \leq \text{ANGLE} \leq 2\Pi)$

PROCEDURE : 1. ANGLE = $\tan^{-1}[\text{Y2-Y1})/(\text{X2-X1})]$. If ANGLE < 0 increment by 2Π.

2. Return.

CF 24/4/78

MODULE : 8.05 VOL

PURPOSE : Draw volume lines for a given link for vehicle and transit volume plots.

INPUT : Subroutine parameters

XB, YB	Starting coordinates for first volume line.
XE, YE	End coordinates for first volume line.
PHIB, PHIE	Angles determining starting and end points of proceding volume lines.
NLINES	Number of volume lines to be drawn for link.
ALPHA	Angle between link and x-axis determined using FUNCTION ANGLE (module 8.04).

PROCEDURE : The area to be blacked in has the structure as shown in Fig 8.05a. The first line joins (X_b, Y_b) to (X_e, Y_e).

At (X_e, Y_e) the pen is moved to (X_e', Y_e') (Fig. 8.05b) where

$$X_e' = X_e + S_2 \cos \Phi_e$$

$$Y_e' = Y_e + S_2 \sin \Phi_e$$

$$S_2 = 0.01/\sin(\alpha - \Phi_e)$$

The perpendicular separation between two consecutive lines is 0.01". A second volume line is then drawn between (X_e', Y_e') and (X_b', Y_b') where

$$X_b' = X_b + S_1 \cos \Phi_b$$

$$Y_b' = Y_b + S_1 \sin \Phi_b$$

$$S_1 = 0.01/\sin(\alpha - \Phi_b)$$

This repetitive procedure is continued until NLINES lines have been drawn.

SYSTEM PLOT ROUTINES : PLOT*

* CDC system plot routine

CF 24/4/78

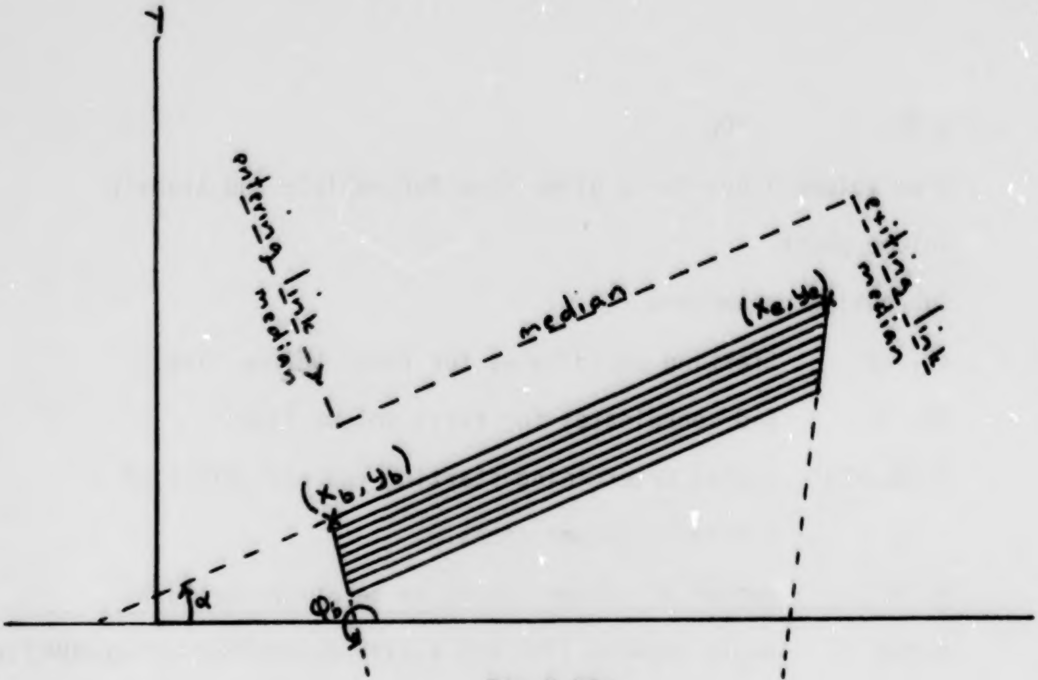

Y
entering ½ 1/k median
median
exiting ½ 1/k median
(xe,ye)
(xb,yb)
α
φb

Fig 8.05a

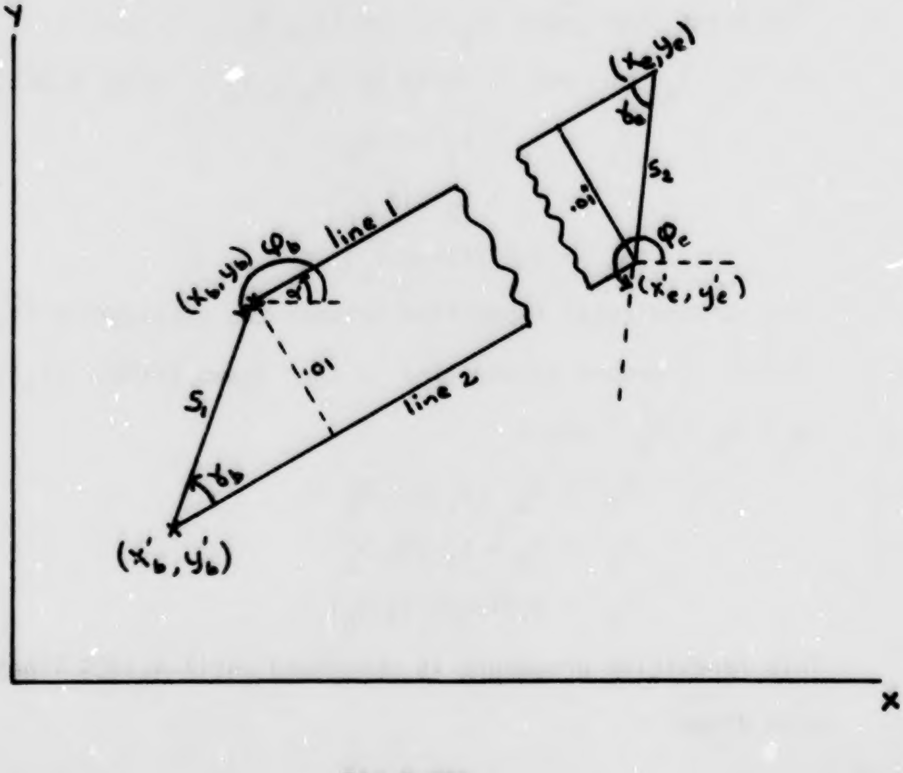

Y
(xe,ye)
γe
S2
.01
φe
(x'e, y'e)
line 1
φb
(xb,yb)
S
.01
line 2
S1
γb
(x'b, y'b)
X

Fig 8.05b

MODULE : 8.06 VOLNUM

PURPOSE : Plots volume numbers above volume lines on a given link for vehicle and transit volume plots.

INPUT : a) Data in COMMON block /GEOM/

WSYM Quotient width/height for a plotted character using the system supplied plot routine SYMBOL.

b) Subroutine parameters

XB, YB	x and y coordinates for starting point of last volume line.
XE, YE	x and y coordinates for end point of last volume line.
ALPHA	Angle between link and x-axis
NUM	For a regular network link,NUM is the traffic volume on that link. For a turning link, NUM is -(traffic volume).

PROCEDURE : If link length too short to plot volume number (< 0.4"), return. Otherwise determine location (X_2, Y_2) for number as shown in Fig. 8.06a.

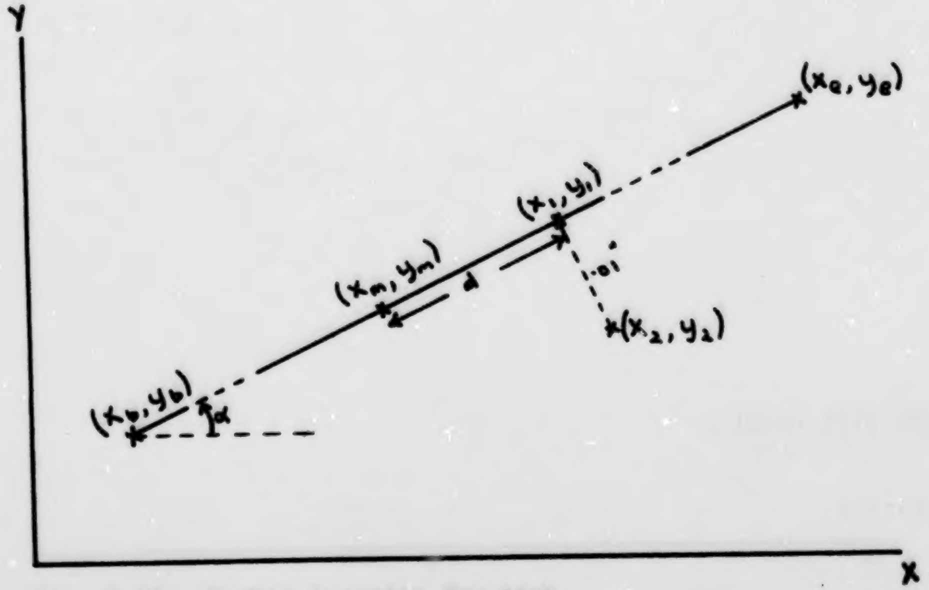


Fig. 8.06a Number location for link.

The number is located at a displacement d from the link midpoint (X_m, Y_m) along the direction of the link and a displacement 0.01", perpendicular to link, from last volume line. Thus

$$X_m = (X_b + X_e)/2$$
$$Y_m = (Y_b + Y_e)/2$$
$$d = 0.03*WSYM*NDG$$

where NDG is the number of digits in the volume number.

$$X_1 = X_m + d\cos\alpha$$
$$Y_1 = Y_m + d\sin\alpha$$
$$X_2 = X_1 + 0.01*\sin\alpha$$
$$Y_2 = Y_1 - 0.01*\cos\alpha$$

For turning links, d is incremented by 0.15" to avoid over printing.

The inclination of the number to the x-axis is $\alpha + \Pi$

SYSTEM PLOT
ROUTINE : NUMBER*

---

* CDC plot routine

MODULE : 8.07 BAR

PURPOSE : Plot volume scale on vehicle and transit volume plots.

INPUT : a) Subroutine parameters:

XU Largest x-coordinate value for nodes in plotted region.

YU Largest y-coordinate value for nodes in plotted region.

b) Data in COMMON block /GEOM/

WSYM Quotient width/height for a plotted character using the system supplied plot routine SYMBOL

VMAX User specified upper limit for link volume

PROCEDURE : The volume scale has the structure as indicated in Fig. 8.07a

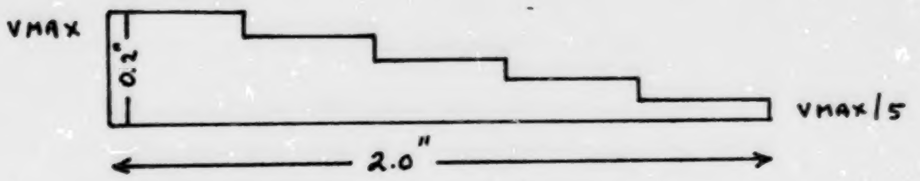


Fig. 8.07a

The plotting order is VMAX, scale and VMAX/5

SYSTEM PLOT ROUTINES : NUMBER, PLOT*

---

* CDC system plot routines

CF 24/4/78

MODULE : 8.09 BLOCK DATA

PURPOSE : Specify unit numbers and some plot parameters which may require user modification. Required by all transit and vehicle plot programs.

VARIABLES SPECIFIED : a) Variables in COMMON block /PLUNIT/

PSPEC	File number of user supplied plot data.
PTITDT	File containing plot title

b) Variables in COMMON block /UNIT/

RLTIVL	File containing predicted link volumes and travel times.
TNET	File containing transit network data.
TFLOW	File containing transit volumes.

c) Variables in COMMON block /GEOM/

WSYM	Quotient width/height for a plotted character using the system supplied routine SYMBOL. *
YMAX	Upper limit for plot height

d) Variables in COMMON block /PARAM/

DELTA,DELTA1	Parameters determining point of intersection of two volume lines as shown in Fig. 8.09a

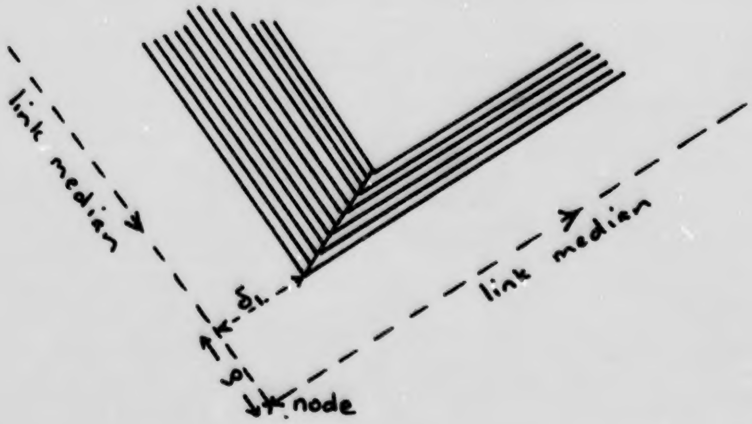


Fig. 8.09a

* CDC System plot routine

CF 24/4/78

MODULE : 8.10 DPARAM

PURPOSE : Determine beginning and end points of link median for vehicle and transit volume plots.

INPUT : a) Data in COMMON blocks /PARAM/ and /GEOM/

DELTA	Perpendicular distance of first volume line from median
WSYM	Quotient width/height for a plotted characted using system supplied plot routine SYMBOL.*

b) Subroutine parameters

X1, Y1	Coordinates of origin node.
X2, Y2	Coordinates of destination node.

OUTPUT : Subroutine parameters

X1, Y1	Starting point of median line
X2, Y2	End point of median line
DIST	Length of median
THETA	Angle between median and x-axis.

PROCEDURE : 1. Calculate length of median line.

2. Find beginning and end points of median, leaving room for node numbers at each end.

3. Return.

---

* CDC plot routine

MODULE : 8.11 DASH

PURPOSE : Plot dashed line between two points

INPUT : Subroutine parameters

X1, Y1	Coordinates of starting point
X2, Y2	Coordinates of end point
DOWN	Length of dash
DIST	Length of line
THETA	Angle of line to x-axis

PROCEDURE : 1. Plot sequence of dashes (length DOWN) and blanks (0.1"), stopping when length DIST has been covered.

2. Return.

SYSTEM PLOT ROUTINES : PLOT *

---

* CDC system plot routine

CF 24/4/78

MODULE : 8.12 CONFGA

PURPOSE : Determine plot parameters X_b, Y_b Φ_b or X_e, Y_e, and Φ_e for link with volume segment as shown in Fig. 8.12a.

INPUT : a) Data in COMMON block /PARAM/

EPS	Arctan (δ_1/δ)
R	$\delta/\cos\varepsilon$

b) Subroutine parameters

ALPHA	Angle of link to x-axis
XO, YO	x and y coordinates of node
U	Sign parameter: +1 for beginning of link, -1 for end of link.

OUTPUT : Subroutine parameters

X	x-coordinate for beginning or end of first volume line
Y	y-coordinate for beginning or end of first volume line
PHI	Φ_b if $U = +1$, Φ_e if $u = -1$

PROCEDURE : The volume segment to be blacked in has the general structure as shown in Fig. 8.05a. CONFGA determines the values of X, Y and PHI for configuration type "A" (Fig. 8.12a) (Xo, Yo) are the node coordinates.

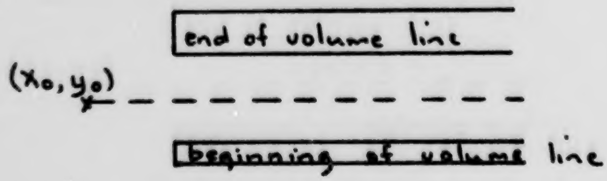
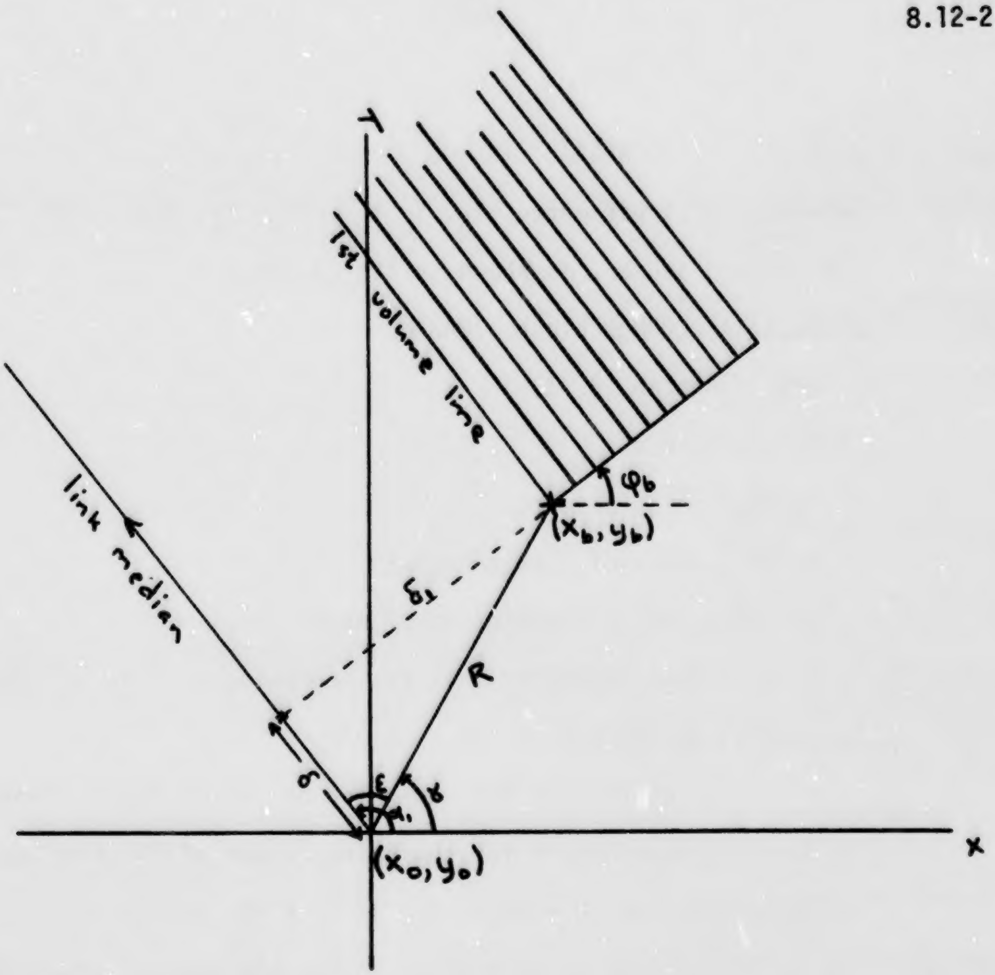


Fig 8.12a Configuration type "A"

X_b, Y_b, Φ_b, X_e, Y_e, and Φ_e are calculated as shown on the next two pages.


Y
1st volume line
link median
δ
δ₁
R
ε
α₁
γ
φb
(xb, yb)
(xo, yo)
x

$\gamma = \alpha_1 - \varepsilon$

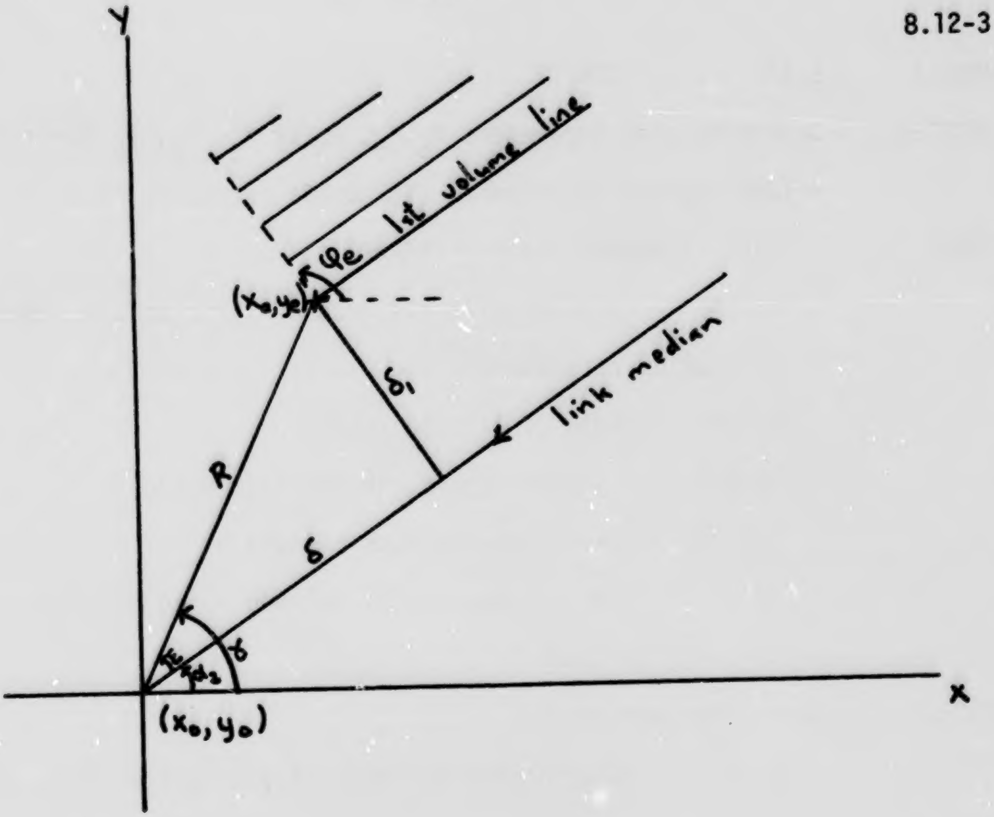
where $\varepsilon = \tan^{-1} (\delta_1/\delta)$

$X_b = X_0 + R \cos\gamma$

$Y_b = Y_o + R \sin\gamma$

$\Phi_b = \alpha_1 - \Pi/2$

Fig. 8.12b. Beginning configuration of type "A".


Y
X
1st volume line
φe
(xe,ye)
δ1
link median
R
δ
γ
ε
α2
(xo,yo)

MODULE : 8.13 CONFGB

PURPOSE : Determine plot parameters X_b,Y_b, Φ_b or X_e, Y_e, Φ_e for link with volume segment as shown in Fig. 8.13a

INPUT : a) Data in COMMON block /PARAM/

DELTA1	Perpendicular distance of first volume line to link median.

b) Subroutine parameters

ALPHA	Angle of link to x-axis
ANGMAX	σ (refer Fig. 8.13b and Fig. 8.13c)
XO, YO	x and y coordinates of node
U	Sign parameter: +1 for beginning of link, -1 for end of link

OUTPUT : Subroutine parameters

X	x-coordinate for beginning or end of first volume line
X	y-coordinate for beginning or end of first volume line
PHI	Φ_b if U = +1, Φ_e if U = -1.

PROCEDURE : The volume segment to be blacked in has the general structure as shown in Fig. 8.05a. CONFGB determines the values of X, Y and PHI for configuration type "B" (Fig. 8.13a) (X_0,Y_0) are the node coordinates.

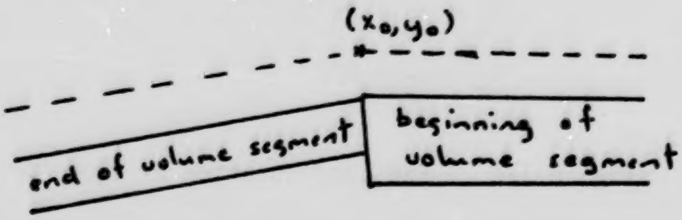
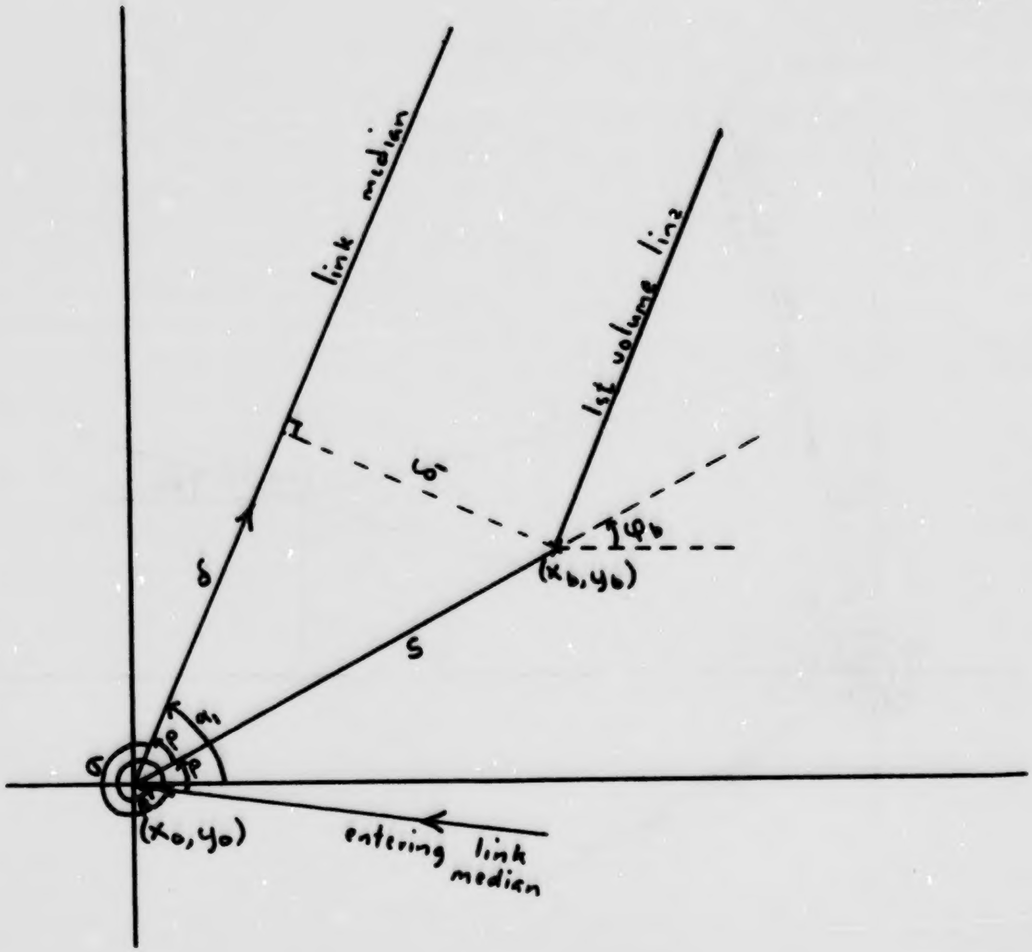


Fig 8.13a

X_b, Y_b, Φ_b, X_e, Y_e, and Φ_e are calculated as shown on the next two pages.

CF 24/4/78


link median
1st volume line
δ
S
ϕ_b
(x_b, y_b)
α_1
ρ
ρ
σ
(x_0, y_0)
entering link median

$\sigma = \beta - \alpha_1$

ANGMAX = σ

$\Phi_b = (2\alpha_1+\sigma)/2 - \Pi$

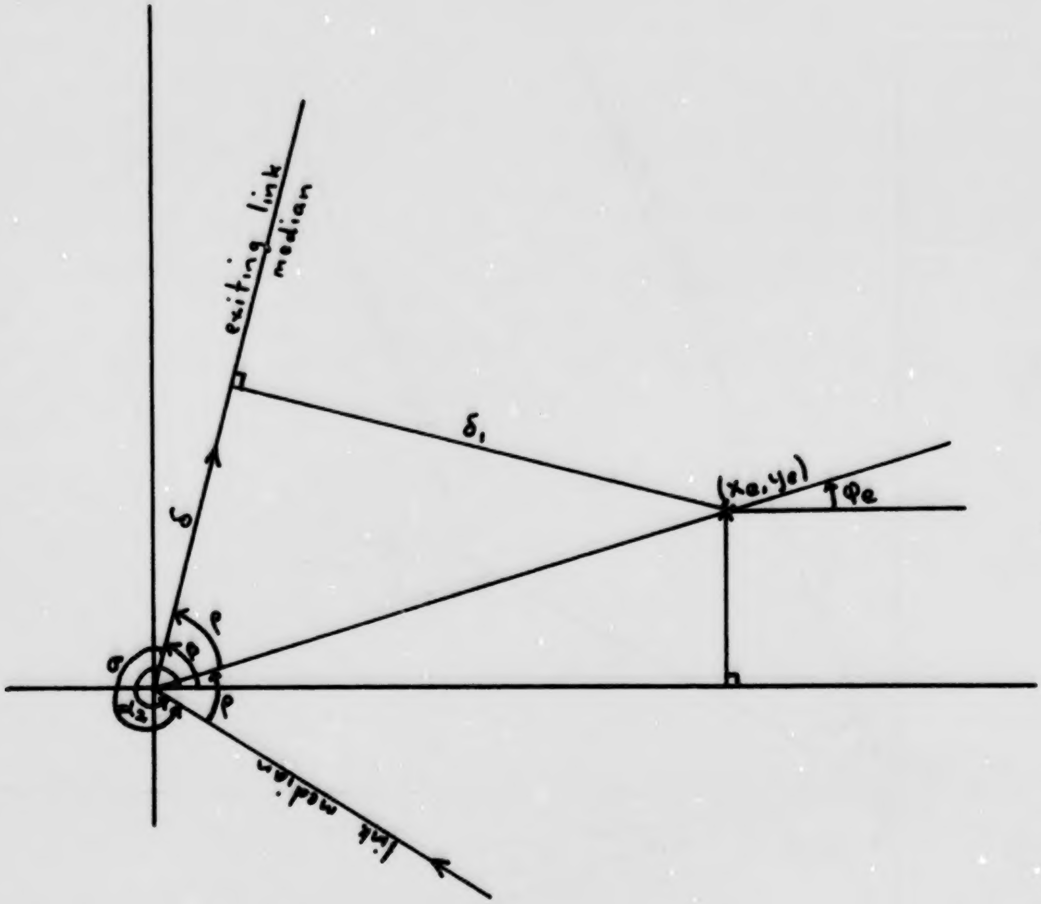
$\rho = \Pi + (\alpha_1-\beta)/2$

$S = \delta_1 / \sin\rho$

$X_b = X_0 + S\cos\Phi_b$

$Y_b = Y_0 + S\sin\Phi_b$

Fig. 8.13b. Beginning configuration of type "B"


exiting link median
δ₁
(xe, ye)
φe
δ
ρ
σ
α
α₂
ρ
link median

$\sigma = \alpha_2 - \beta$ $\quad\quad$ σ = ANGMAX

$\rho = \Pi - \sigma/2$

$S = \delta_1 / \sin(\Pi - \sigma/2)$

$\Phi_e = (2\alpha_2 - \sigma)/2 - \Pi$

$X_e = X_0 + S \cos\Phi_e$

$Y_e = Y_0 + S \sin\Phi_e$

Fig. 8.13c. End configuration of type "B"

MODULE : 8.14 CONFGC

PURPOSE : Determine plot parameters X_b, Y_b, Φ_b or X_e, Y_e and Φ_e for link with volume segment as shown in Fig. 8.14a.

INPUT : a) Data in COMMON block /PARAM/

DELTA1	Perpendicular distance of first volume line to link median
NV	Volume represented by 0.01"

b) Subroutine parameters

ALPHA	Angle of link to x-axis
ANGMAX	σ (as shown in FIGS. 8.14b and 8.14c)
XO, YO	x and y coordinates of node
U	Sign parameter: +1 for beginning of link, -1 for end of link
NLINES	Number of volume lines
LVLENT	For U = +1, LVLENT is volume on first link (in clockwise sense) entering node at (XO, YO). For U = -1, LVLENT is volume on first link (in anti-clockwise sense) exiting node at (XO, YO).

OUTPUT : Subroutine parameters

X	x-coordinate for beginning or end of first volume line.
Y	y-coordinate for beginning or end of first volume line.
PHI	Φ_b if $U = +1$, Φ_e if $U = -1$

PROCEDURE : The volume segment to be blacked in has the general structure as shown in Fig. 8.05a. CONFGC determines the values of X, Y and PHI for configuration type "C" (Fig. 8.14a). (X_0, Y_0) are the node coordinates.

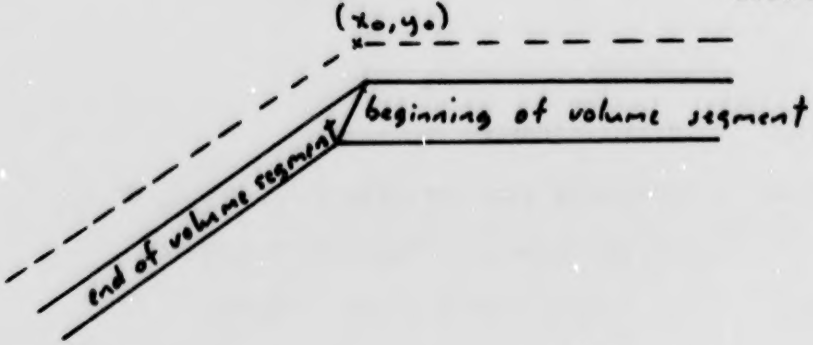
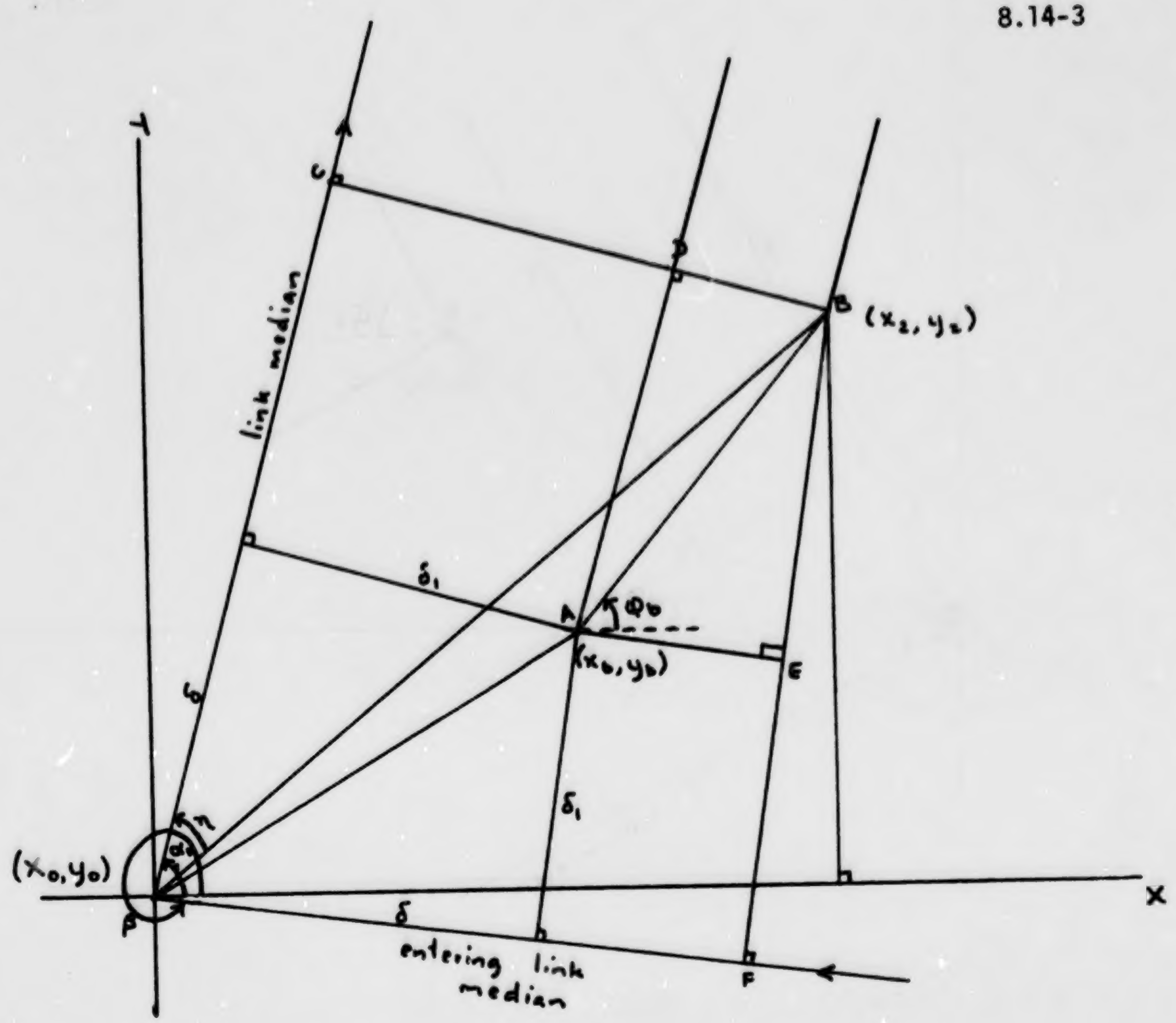

(x_0, y_0)
beginning of volume segment
end of volume segment

Fig. 8.14a

X_b, Y_b, Φ_b and X_e, Y_e, Φ_e are calculated as shown on the next two pages.


T
C
D
B (x2, y2)
link median
δ1
A
Φb
(xb, yb)
E
δ
η
αb
α1
(x0, y0)
β
X
δ
δ1
entering link median
F

$\sigma = \beta - \alpha \qquad \sigma = \text{ANGMAX}$

$S = OA = \delta_1 / \sin(\Pi - \sigma/2)$

$X_b = X_0 + S \cos[(2\alpha_1 + \sigma)/2 - \Pi]$

$Y_b = Y_0 + S \sin[(2\alpha_1 + \sigma)/2 - \Pi]$

$\eta = < BOC$

$\Omega_1 = DB, \ \Omega_2 = BE$

$$\eta = \tan^{-1} \frac{[(\Omega_1 + \delta_1) \sin(-\sigma)]}{\Omega_2 + \delta_1 + (\delta_1 + \Omega_1) \cos(-\sigma)}$$

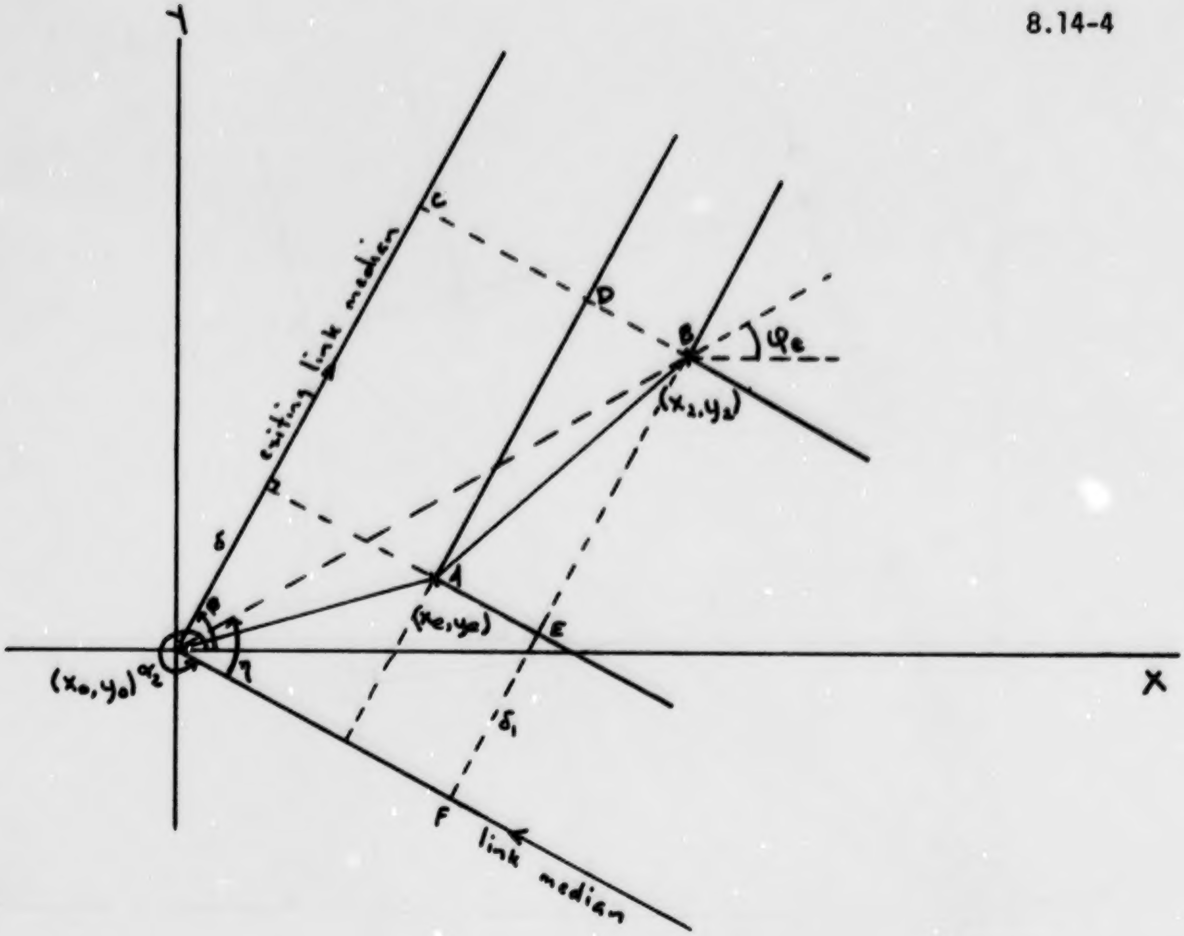
$S' = OB = (\Omega_1 + \delta_1)/\sin\eta$

$X_2 = X_0 + S' \cos(\alpha_1 - \eta)$

$Y_2 = Y_0 + S' \sin(\alpha_1 - \eta)$

$\Phi_b = \tan^{-1} [(Y_2 - Y_b)/(X_2 - X_b)]$

Fig. 8.14b. Beginning configuration type "C"


Y
C
existing link median
D
B
φ_e
(x_2, y_2)
δ
A
θ
(x_e, y_e)
E
(x_o, y_o)
α_2
η
X
δ_1
F
link median

$S = OA = \delta_1/\sin(\Pi - \sigma/2)$ $\sigma = \text{ANGMAX}$

$= \alpha_2 - \beta$

$X_b = X_0 + S \cos[(2\alpha_2 - \sigma)/2 - \Pi]$

$Y_b = Y_0 + S \sin[(2\alpha_2 - \sigma)/2 - \Pi]$

$\eta = < BOF$

$\Omega_1 = BE, \quad \Omega_2 = BD$

$$\eta = \tan^{-1} \frac{(\delta_1 + \Omega_1)\sin(-\sigma)}{(\Omega_1 + \delta_1)\cos(-\sigma) + (\Omega_2 + \delta_1)}$$

$S' = OB = (\Omega_1 + \delta_1)/\sin\eta$

$X_2 = X_0 + S' \cos(\alpha_2 + \eta)$

$Y_2 = Y_0 + S' \sin(\alpha_2 + \eta)$

$\Phi_e = \tan^{-1}[(Y_2 - Y_e)/(X_2 - X_e)]$

Fig. 8.14c. End configuration type "C"

MODULE : 8.15 CONFGD

PURPOSE : Determine plot parameters X_b, Y_b, Φ_b or X_e, Y_e and Φ_e for link with volume segment as shown in Fig. 8.15a.

INPUT : a) Data in COMMON block /PARAM/

DELTA1	Perpendicular distance of first volume line to link median.

b) Subroutine parameters

ALPHA	Angle of link to x-axis
BETA	β as shown in Figs. 8.15b and 8.15c.
XO, YO	x and y coordinates of node
U	Sign parameter: +1 for beginning of link, -1 for end of link.

OUTPUT : Subroutine parameters

X	x-coordinate for beginning or end of first volume line
Y	y-coordinate for beginning or end of first volume line.

PROCEDURE : The volume segment to be blacked in has the general structure as shown in Fig. 8.05a. CONFGD determines the values of X and Y for configuration type "D" (Fig. 8.15a). (X_0, Y_0) are the node coordinates.

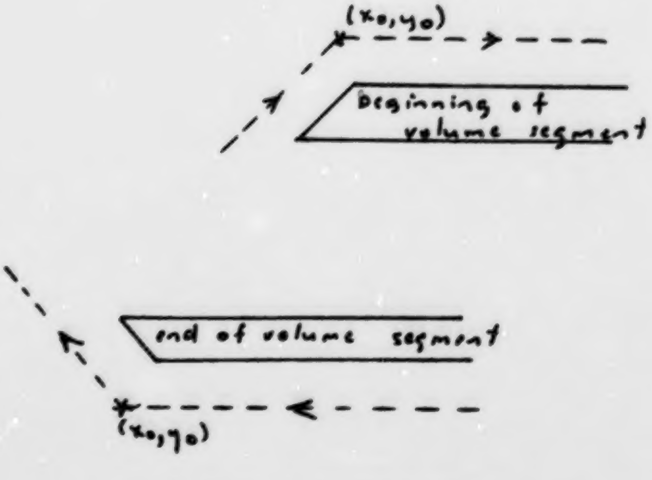
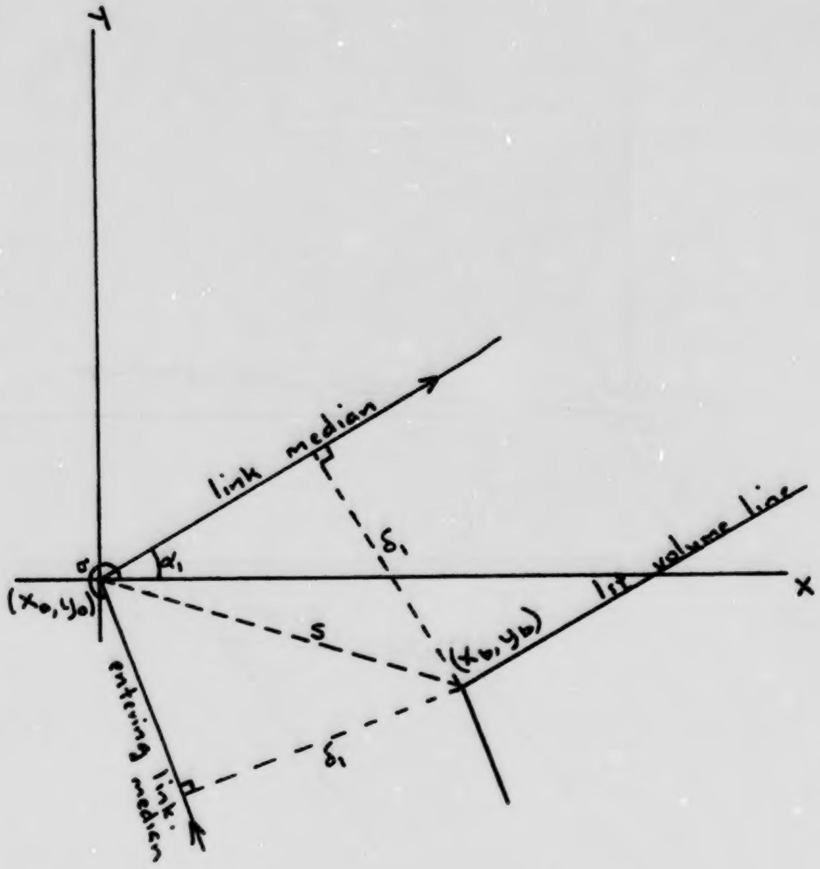


Fig. 8.15a Configuration type "D"

X_b, Y_b, X_e and Y_e are calculated as shown on the next two pages. Φ_b and Φ_e are simply the angles between the intersecting links and are calculated in AVLINE (automobile volume plot) and TVLINE (transit volume plot).


Y
link median
δ1
1st volume line
X
σ
α1
(x0,y0)
S
(xb,yb)
entering link median
δ1

$$\Phi_b = \sigma$$

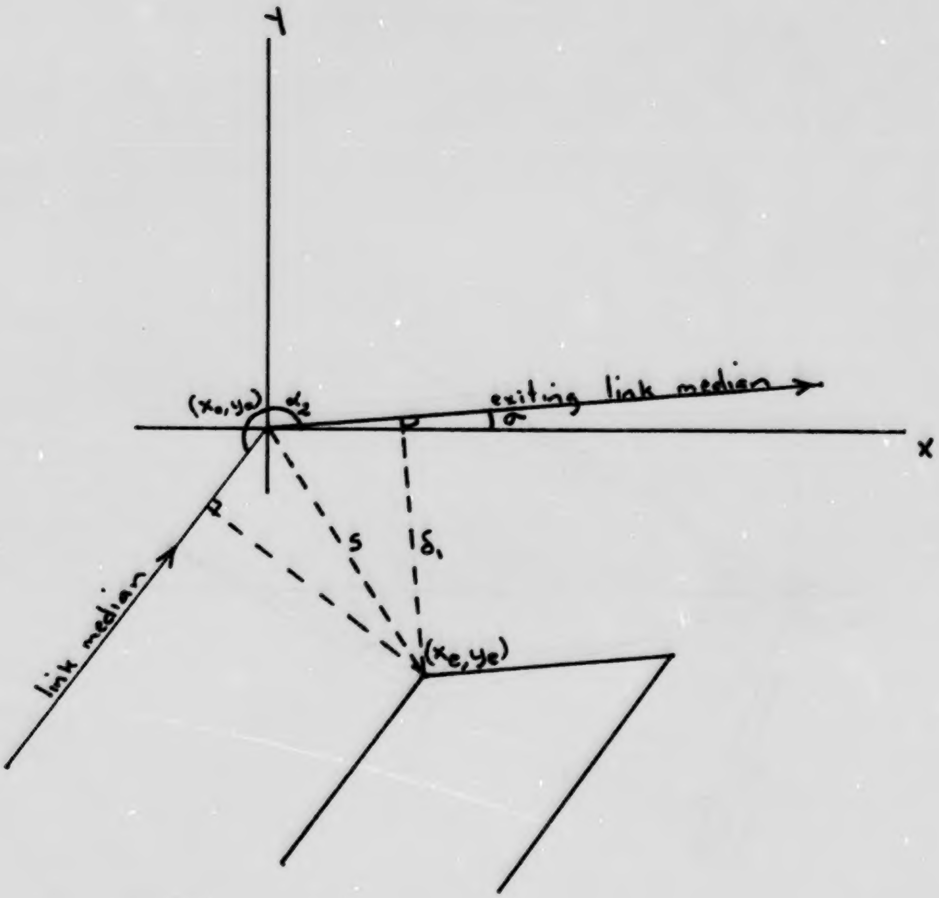
$$\beta = \sigma - \alpha_1$$

$$S = \delta_1 / \sin(\Pi - \beta/2)$$

$$X_b = X_0 + S \cos(\alpha_1 + \beta/2 - \Pi)$$

$$Y_b = Y_0 + S \sin(\alpha_1 + \beta/2 - \Pi)$$

Fig. 8.15b. Beginning configuration of type "D"


Y
X
(x_o,y_o)
α_2
exiting link median
σ
S
δ_1
link median
(x_e,y_e)

$\Phi_e = \sigma$

$\beta = \alpha_2 - \sigma$

$S = \delta_1 / \sin(\Pi - \beta/2)$

$X_b = X_0 + S \cos(\alpha_2 - \beta/2 + \Pi)$

$Y_b = Y_0 + S \sin(\alpha_2 - \beta/2 + \Pi)$

Fig. 8.15c. End configuration of type "D"

MODULE : 8.21 AUNETP

PURPOSE : Automobile network plot, MAIN program.

INPUT : Data is accessed through routines OPBAN, DINAUN and a BLOCK DATA subprogram.

a) Data from OPBAN

NNODE	Number of vehicle nodes
XINFIN	Large real number

b) Data from DINAUN

ANGROT	Angle in counterclockwise sense, through which plot is rotated with respect to the geo-coded coordinate system.
NX1, NX2	Node numbers giving lower and upper bounds along x-axis after rotation of plot axes through ANGROT.
NY1, NY2	Node numbers giving lower and upper bounds along y-axis after rotation of plot axes through ANGROT.
XNO(K)	x-coordinate of node K (internal node number).
YNO(K)	y-coordinate of node K (internal node number).

c) Data from BLOCK DATA subprogram (module 8.09)

YMAX	Maximum plot range, y-axis (inches).

PROCEDURE : In the following description, names in left hand column are subroutines which contain code for the section described to the right.

1.		Initialize OK to TRUE.
2.	OPBAN	Initialize data bank; OK set to FALSE if errors encountered.
3.		If OK is FALSE, exit; otherwise proceed to next step.
4.	NETWRK	Read in network description. Set OK to FALSE if errors encountered.
5.		If OK is FALSE, exit; otherwise proceed to next step.
6.	DINAUN	Read in user supplied plot data (module 8.22); store node coordinates in vectors XNO and YNO.

7.	LIMITE[+]	Request maximum plot height YMAX
8.		Rotate plot axes in anti clockwise sense through ANGROT; transform coordinates so that the minimum x and y coordinate values are 0.
9.		Check that node NY2 with maximum y-coordinate will fit on plot. If negative, print error message and exit.
10.	NETPLT	Plot network (module 8.23).
11.		Stop.

[+] CDC system routine

MODULE : 8.22 DINAUN

PURPOSE : Read in user supplied data for vehicle network plot.

INPUT : a) Input file

File name : PSPEC

Type : Direct entry unput file

Status : Required

Contents : M1, M2 - external node numbers of nodes defining lower and upper boundaries of plot along x-axis after rotation of plot axes through ANGROT.

N1, N2 - external node numbers of nodes defining lower and upper boundaries of plot along y-axis after rotation of plot axes through ANGROT.

I1, I2, J1, J2 - external node numbers of nodes defining a rectangle internal to plot area to be omitted from plot. I1 and I2 define lower and upper bounds of rectangle along x-axis (after rotation); J1 and J2 define lower and upper bounds of rectangle along y-axis (after rotation) if no segment is to be omitted, I1 is 0.

SCALE - desired scale of plot (ft./in.)

ANGROT - angle (degrees) in counterclockwise sense through which plot axes are rotated with respect to axes in which nodes were geo-coded.

ANGNOR - angle (degrees) between x-axis and north, counterclockwise sense, in system of reference used for geo-coding.

Sequence : Sequential

Source : User specified

b) Data in COMMON areas /NETSIZ/ and /PLUNIT/

NNODE	Number of vehicle nodes in network.
PSPEC	File number of input file PSPEC, specified in BLOCK DATA (module 8.09).
INFINI	Large positive integer.
XINFIN	Large real number

OUTPUT : Data in COMMON areas /NOEUD/, /BOUND/ and /INTEX/

XNO(K)	x-coordinate of node K.
YNO(K)	y-coordinate of node K.
EXTN(K)	External node number of node K.
NX1,NX2 NY1,NY2	Internal node numbers of M1, M2, N1 and N2 respectively.
IX1,IX2 IY1,IY2	Internal node numbers of I1, I2, J1 and J2 respectively.

PROCEDURE :

1. Rewind file PSPEC.
2. Read in data from PSPEC.
3. Determine number of coordinate units per mile using routine GTSCAL.
4. Retrieve node coordinates from data bank using GTNODE.
5. Determine internal node numbers of boundary nodes.
6. Return.

MODULE : 8.23 NETPLT

PURPOSE : Create subnetwork of vehicle network links to be plotted for vehicle network plot.

INPUT : a) Data in COMMON blocks /NOEUD/, /BOUND/, /GEOM/, /NETSIZ/ and /RNET1/

XNO(K)	x-coordinate for node K.
YNO(K)	y-coordinate for node K.
IX1, IX2	Node numbers (internal) defining lower and upper bounds along x-axis of rectangle, internal to plot area, to be omitted from plot.
IY1, IY2	Node numbers (internal) defining lower and upper bounds along y-axis of rectangle, internal to plot area, to be omitted from plot.
SCALE	Desired scale of plot (ft./in.)
NCENT	Number of zone centroids.
NNODE	Number of vehicle network nodes.
DIST(L)	Length of link L.
LTYP(L)	Link type of link L.
NOLAB(K)	Location of first occurence of node K in link table.
ND(L)	Destination node of link L.

b) Subroutine parameters

XU	Range of plot along x-axis.
YU	Range of plot along y-axis.

OUTPUT : Data in COMMON blocks /NOEUD/, /NETSIZ/ and /RNET1/

NEXIT(K)	Number of links in plot area exiting node K.
NLINK	Number of links in plot area.
NO(L)	Origin node for link L in sub-link table for plot area.
ND(L)	Destination node for link L in sub-link table for plot area.

NOLAB(K) Link number of first occurrence of node K in sub-link table.

PROCEDURE : In the following description, names in left hand column are routines which contain code for the section described to the right.

1.	Create sublink table of links in plot area. For each such link compare calculated link length with measured length using routine ERRDST (module 8.25). If discrepency greater than 20% print error message.
2. PTITLE	Plot heading on plot (module 8.01).
3. HEADNG	Plot distance scale and direction indicator (module 8.02).
4. ANALGO	Determine order in which links are plotted; access plot routines which plot node numbers and link medians.
5. ENDPLT[+]	Terminate plot file.
6.	Return.

[+] CDC system plot routine

MODULE : 8.24 ANALGO

PURPOSE : Determine order in which links are plotted, vehicle network plot; access plot routines.

INPUT : Data in COMMON blocks /NOEUD/, /INTEX/, /NETSIZ/ and /RNET1/

NEXIT(K)	Number of links in plot area exiting node K.
EXTN(K)	External node number of node K.
NLINK	Number of links in plot area.
NNODE	Number of vehicle network nodes.
NO(L)	Origin node of link L in sub-link table of nodes in plot area.
ND(L)	Destination node of link L in sub-link table of nodes in plot area.
NOLAB(K)	Location of first occurrence of node K in sub-link table.

PROCEDURE : In the following description, names in the left hand column are subroutines which contain code for the section described to the right.

1.	Initialize NPLOT(K) to FALSE, K=1,NNODE. Set LSTART=1.
2.	Find first unplotted link in link table beginning with link number LSTART. If no unplotted links are found, return. Otherwise set LINK and LSTART equal to link number of first unplotted link.
3. ANNODE	Plot origin node number NP1 of LINK if not already plotted. Set NPLOT(NP1) to TRUE.
4.	Set NP2 equal to destination node number of link number LINK: check if median of link has been plotted. If affirmative go to step 6.
5. LINKPL	Plot link median. If link is one-way, plot arrow indicating link direction.

6. ANNODE Plot node number NP2 if not already plotted. Set NPLOT(NP2) to TRUE.

7. Increment NEXIT(NP1) by 100 (Number of links exiting NP1 which have been plotted is given by quotient NEXIT(NP1)/100).

8. If some links exiting NP1 remain to be plotted go to step 9. Otherwise remove NP1 from cumulative table NTABLE. This vector accumulates nodes encountered from starting point LSTART. When all links exiting a given node have been plotted, the node number is removed from the table.

9. Investigate node NP2 as starting point for next link plot. Set NP1=NP2, enter NP2 on NTABLE.

10. Determine if all links exiting NP1 have been plotted. If negative go to step 12.

11. Locate first node in NTABLE having an unplotted exiting link. If no links are found return to step 2. Otherwise set NP1 equal to node number.

12. Find first unplotted link from NP1. Return to step 3.

MODULE : 8.25 ERRDST

PURPOSE : Compare computed and measured link lengths, automobile network plot.

INPUT : a) Data in COMMON block /NOEUD/

XNO(K)	x-coordinate of node K.
YNO(K)	y-coordinate of node K.

b) Subroutine parameters

XDIST	Measured link length supplied by user.
IFLAG	Flag set to 0 on entry to subroutine
FACTOR	Factor required to convert computed link length into same units as measured link length.
N1	Origin node of link.
N2	Destination node of link.

OUTPUT : Subroutine parameters

IFLAG	Flag set to 1 if discrepency between measured and calculated link length is greater than 20%.
XDIST	Calculated link length,

PROCEDURE : 1. Compute link length from node coordinates. Convert length into same units (miles) as the measured values.

2. Compare lengths; if difference is greater than 20% set IFLAG to 1.

3. Return.

CF 11/5/78

MODULE : 8.26 ANNODE

PURPOSE : Plot node number for vehicle network plot.

INPUT : a) Data in COMMON blocks /NOEUD/, /INTEX/ and /GEOM/

XNO(K)	x-coordinate of node K.
YNO(K)	y-coordinate of node K.
EXTN(K)	External node number of node K.
ANGROT	Angle through which plot axes are rotated with respect to system of reference of coordinates in data bank.
CT, ST	Cosine and sine of ANGROT.

b) Subroutine parameter

NP Node number (internal) of node to be plotted.

PROCEDURE : 1. Determine external number for node NP.

2. Plot circle centred at node location using system routine SYMBOL[+].

3. Determine number of digits in external node number.

4. Plot number using system plot routine NUMBER[+].

5. Return.

SYSTEM PLOT
ROUTINES: SYMBOL, NUMBER.

[+] CDC system plot routine

CF 11/5/78

MODULE : 8.27 LINKPL

PURPOSE : Plot link median, vehicle network plot.

INPUT : a) Data in COMMON blocks /NOEUD/ and /NETSIZ/

XNO(K) x-coordinate, node K.

YNO(K) y-coordinate, node K.

NCENT Number of zone centroids.

b) Subroutine parameters

NP1 Origin node of link.

NP2 Destination node of link.

IARR 0 if NP1 - NP2 is a two-way link;

1 if one-way link.

PROCEDURE :

1. Compute starting and end points on plot for link median.
2. If link is a centroid connector, go to step 4.
3. Draw straight line joining NP1 and NP2. If link is one-way, plot arrow indicating direction. Return.
4. Draw dashed line between NP1 and NP2 using routine DASH (module 8.11) and return.

CF 11/5/78

MODULE : 8.41 AUVOLP

PURPOSE : Automobile volume plot, MAIN program

INPUT : Data is accessed through routines OPBAN, DINAUV and a BLOCK DATA subprogram.

a) Data from OPBAN

NNODE	Number of vehicle nodes.
XINFIN	Large real number.

b) Data from DINAUV (module 8.42)

VMAX	Upper bound for vehicle link volumes.
ANGROT	Angle, in counterclockwise sense, through which plot is rotated with respect to the geo-coded coordinate system.
NX1, NX2	Node numbers giving lower and upper bounds along x-axis after coordinates are rotated through ANGROT.
NY1, NY2	Node numbers giving lower and upper bounds along y-axis after coordinates are rotated through ANGROT.
XNO(K)	x-coordinate of node K (internal node number).
YNO(K)	y-coordinate of node K (internal node number).

c) Data from BLOCK DATA subprogram (module 8.09).

DELTA, DELTA1	Parameters determining the points of intersection of volumes lines for two connected links.
YMAX	Maximum permissible plot height (inches).

PROCEDURE : In the following description, names in left hand column are subroutines which contain code for the section described to the right.

1. Initialize OK to TRUE.
2. OPBAN Initialize data bank; set OK to FALSE if any errors encountered.

3.	If OK is TRUE go to step 4; otherwise exit.
4. NETWRK	Read in network description. Set OK to FALSE if errors encountered.
5.	if OK is FALSE, exit. Otherwise proceed to next step.
6. DINAUV	Read in user supplied plot data (module 8.42). Store node coordinates in vectors XNO and YNO.
7. LIMITE[+]	Request maximum plot height YMAX.
8.	Determine values for plot parameters EPS, R and NV. EPS and R determine the distance of link volume lines from the nodes. NV designates the vehicle volume represented on the plot by 0.01".
9. TABLE	Store vehicle link volumes for links in plot area in LVOL. If link table from the solution file is not compatible with the network description obtained in step 3, exit.
10.	Rotate axes through ANGROT. Transform coordinates so that the minimum x and y coordinate values are 0.
11.	Check that node NY2 with the maximum y-coordinate value will fit on plot. If negative, print error message and exit.
12. VOLPLT	Plot network and link volumes.
13.	Stop.

[+] CDC system routine

MODULE : 8.42 DINAUV

PURPOSE : Read in user supplied data for auto volume plot.

INPUT : a) Input file

File name : PSPEC

Type : Direct entry input file

Status : Required

Contents : M1, M2 - external node numbers of nodes defining lower and upper boundaries of plot along x-axis after rotation of plot axes through ANGROT.

N1, N2 - external node numbers of nodes defining lower and upper boundaries of plot along y-axis after rotation of plot through ANGROT.

I1, I2, J1, J2 - external node numbers of nodes defining a rectangle internal to plot area, to be omitted from plot. I1 and I2 define lower and upper bounds of rectangle along x-axis (after rotation); J1 and J2 define lower and upper bounds of rectangle along y-axis (after rotation). If no segment is to be omitted, I1 is 0.

SCALE - desired scale of plot (ft./in.).

ANGROT - angle in counterclockwise sense through which plot axes are rotated with respect to axes in which nodes were geo-coded.

ANGNOR - angle (degrees) between x-axis and north, counterclockwise sense, in system of reference used for geocoding.

VMAX - upper limit for link volume.

Sequence : Sequential.

Source : User specified.

b) Data in COMMON areas /NETSIZ/, /PLUNIT/ and /CONST/

NNODE	Number of vehicle nodes in network
PSPEC	File number of input file PSPEC designated in BLOCK DATA (module 8.09).
INFINI	Large positive integer.
XINFIN	Large real number.

OUTPUT : Data in COMMON areas /NOEUD/, /BOUND/ and /INTEX/

XNO(K)	x-coordinate of node K.
YNO(K)	y-coordinate of node K.
EXTN(K)	External node number of node K.
NX1, NX2 NY1, NY2	Internal node numbers of M1, M2, N1 and N2 respectively.
IX1, IX2 IY1, IY2	Internal node numbers of I1, I2, J1 and J2 respectively

PROCEDURE : 1. Rewind file PSPEC

2. Read in data from PSPEC

3. Determine UNIT which specifies number of coordinate units per mile.

4. Access coordinates from data bank using routine GTNODE

5. Determine internal numbers of nodes defining boundaries using routine PTNODE.

6. Return.

MODULE : 8.43 TABLE

PURPOSE : Read in solution file for vehicle volume plot.

INPUT : a) Data in COMMON areas /NETSIZ/ and /RNET1/

NLINK	Number of links in network.
NNODE	Number of nodes in network.
LTYPE(L)	Link type of link L.
NOLAB(K)	Location of first occurrence of node K in link table.
ND(L)	Destination node of link L.

b) Input file

File name : RLTIVL

Type : Intermediate file

Status : Required

Contents : Predicted auto volume and travel time for each link in road network.

Source : Output from 6.80 RASSAV.

OUTPUT : a) Data in COMMON areas /RNET1/ and /REP/

LVOL(L)	Vehicle volume on link L (stored in DIST which is equivalenced to LVOL).
TITLE	Title of solution file.
TDAT	Date solution file obtained.

b) Subroutine parameters

OK	TRUE if solution file link table is compatible with network description. FALSE if solution file is unusable.

PROCEDURE : 1. Read and print identification record from RLTIVL.

2. Initialise OK to TRUE, IFOUND(L) to FALSE and LVOL(L) to 0, L=1,NLINK.

3. Read a link record from RLTIVL; if end of file is reached go to step 7.

4. Convert link origin and destination node numbers to internal numbers.
5. Check that current link L is in network. If affirmative go to next step. Otherwise print a message and if link volume is non-zero, set OK to FALSE. Return to step 3.
6. Store volume in LVOL(L). Set IFOUND(L) to TRUE and return to step 3.
7. Check that a solution value was found for all links in network. For each link missing from solution file, print message.
8. Rewind RLTIVL and return.

MODULE : 8.44 VOLPLT

PURPOSE : Determine links in plot area,access plot routines for vehicle volume plot.

INPUT : a) Data in COMMON blocks /NOEUD/, /BOUND/, /NETSIZ/, /RNET1/ and /CONST/

XNO(K)	x-coordinate of node K.
YNO(K)	y-coordinate of node K.
IX1,IX2 IY1,IY2	Internal node numbers of nodes defining region interior to the plot area to be omitted from plot. Link I-J is omitted from plot if $XNO(IX1) < XNO(K) < XNO(IX2)$ and $YNO(IY1) < YNO(K) < YNO(IY2)$ for K = I or J.
NNODE	Number of vehicle nodes
LTYP(L)	Link type of link L.
NOLAB(K)	Location of first occurrence of node K in link table.
ND(L)	Destination node of link L.
LVOL(L)	Predicted volume on link L.
XINFIN	Large real positive number.

b) Subroutine parameters

XU, YU	Defines range of plot. Link I-J is plotted if $0 < XNO(K) < XU$, $0 < YNO(K) < YU$,for K = I or J,and link is not internal to rectangle defined by nodes IX1, IX2, IY1 and IY2.

OUTPUT : Data in COMMON areas /NOEUD/, /NETSIZ/, /WORKA/ and /RNET1/

NEXIT(K)	Number of links in plot area exiting node K.
NLINK	Number of links in plot area.
LVOL(L)	Predicted volume of link L.
NO(L)	Origin node of link L.
ND(L)	Destination node of link L.
NOLAB(K)	Location of first occurence of node K in link table.

PROCEDURE : In the following description, names in the left hand column are subroutines which contain the code for the section described to the right.

1.	Create sublink table of links in plot area. If only one node of link L is in plot area, set LVOL(L) to $-\infty$. If link L is a turning link, set LVOL(L) to -LVOL(L). (User supplied routine TURLIN is required. TURLIN(L) is TRUE if link L is a turning link; otherwise FALSE).
2. PTITLE	Plot heading (module 8.01).
3. HEADNG	Plot distance scale and direction indicator (module 8.02).
4. BAR	Plot volume scale (module 8.07).
5. AVALGO	Determine order in which links are plotted; access plot routines (module 8.45).
6. ENDPLT+	Terminate plot file.
7.	Return.

+CDC system plot routine

MODULE : 8.45 AVALGO

PURPOSE : Determine order in which links are plotted for vehicle volume plot, access plot routines

INPUT : a) Data in COMMON blocks /NOEUD/, /WORKA/, /NETSIZ/ and /RNET1/

NEXIT(K)	Number of links exiting node K.
NO(L)	Origin node of link L.
LVOL(L)	Predicted volume on link L. If L is a turning link, LVOL(L) = -(predicted volume).
NOLAB(K)	Location of first occurrence of node K in link table.
ND(L)	Destination node of link L.
NCENT	Number of zone centroids.
NNODE	Number of vehicle nodes.
NLINK	Number of links in plot area.
XNO(K)	x-coordinate of node K.
YNO(K)	y-coordinate of node K.

PROCEDURE : In the following description, names in the left hand column are subroutines which contain code for the section described to the right.

1. Initialize NPLOT(K) to FALSE,(K=1, NNODE); LSTART to 1.

2. Find first unplotted link in link table starting at location LSTART. If all links have been plotted, return. Otherwise set LINK and LSTART equal to link number of first unplotted link.

3. AVNODE Plot origin node NP1 of link number LINK if not already plotted (module 8.46). Set NPLOT(NP1) to TRUE.

4. If link median has already been plotted go to step 7. Otherwise go to next step.

5. PARAM Determine beginning and end points for link median (module 8.10).

6. DASH Draw dashed line along link median (module 8.11).

7. AVNODE Plot destination node NP2 of current link if not already plotted (module 8.46). Set NPLOT(NP2) to TRUE.

8. AVLINE Plot volume lines for current link (module 8.47).

9. Increment NEXIT(NP1) by 100 (number of links exiting NP1 which have been plotted is given by the quotient NEXIT(NP1)/100).

10. If some links exiting NP1 remain to be plotted go to step 11. Otherwise remove NP1 from cumulative table NTABLE. This vector accumulates nodes encountered from starting point LSTART. When all links exiting a given node have been plotted it is removed from the table.

11. Investigate NP2 as starting point for next link plot. Set NP1 = NP2. If no links have been plotted from NP2, add to NTABLE.

12. Determine if all links from NP1 have been plotted. If negative go to step 14. Otherwise go to next step.

13. Locate first node in NTABLE having an unplotted exiting link. If no links are found, return to step 2. Otherwise set NP1 equal to node number.

14. Find first unplotted link exiting NP1. If unplotted link traverses plot boundary (i.e. if $LVOL(L)=-\infty$), increment NEXIT(NP1) by 100 and continue search. If no exiting link found return to step 13. Otherwise set LINK equal to link number and go to step 4.

MODULE : 8.46 AVNODE

PURPOSE : Plot node number for vehicle volume plot.

INPUT : a) Data in COMMON blocks /NOEUD/ and /INTEX/

XNO(K)	x-coordinate for node K
YNO(K)	y-coordinate for node K
EXTN(K)	external node number of node K

b) Subroutine parameter

NP Node to be plotted

PROCEDURE : 1. Determine location of node and number of digits in node number.

2. Plot node number using CDC system plot routine NUMBER.

SYSTEM PLOT
ROUTINES: NUMBER

MODULE : 8.47 AVLINE

PURPOSE : Determine structure of link volume segment, auto volume plot, for a given link; access volume plot routine.

INPUT : a) Data in COMMON blocks /NOEUD/, /PARAM/, /WORKA/, /NETSIZ/ and /RNET1/

XNO(K)	x-coordinate of node K
YNO(K)	y-coordinate of node K
NV	Volume represented by 0.01".
LVOL(L)	Predicted volume on link L.
NO(L)	Origin node of link L.
ND(L)	Destination node of link L.
NCENT	Number of zone centroids.
NLINK	Number of links in plot area.
NOLAB(K)	Location of first occurrence of node K in link table.

b) Subroutine parameters

NP1	Origin node of current link.
NP2	Destination node of current link.
LINK	Current link.

PROCEDURE : The beginning and end sections of a link volume segment may each have one of four different configurations as shown by Figures 8.47a - 8.47d

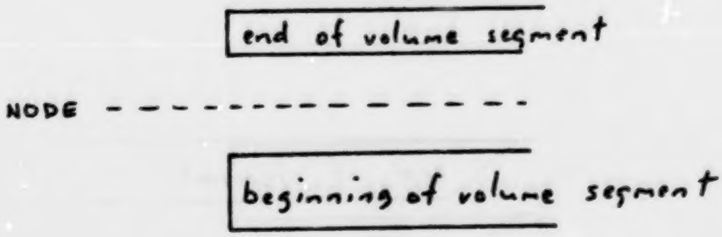


Fig. 8.47a. Configuration type "A"

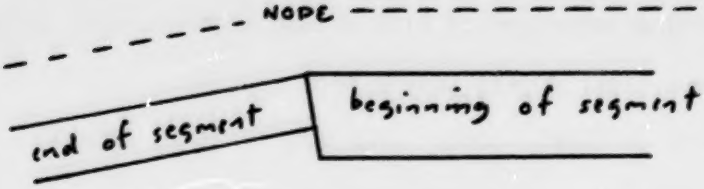

NODE
end of segment
beginning of segment

Fig. 8.47b. Configuration type "B"

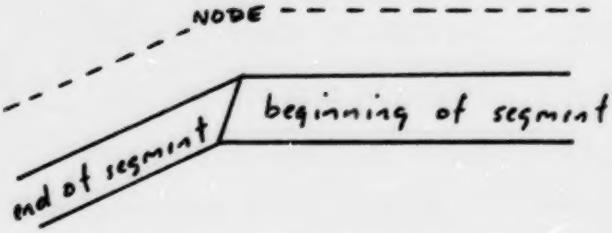

NODE
end of segment
beginning of segment

Fig. 8.47c. Configuration type "C"

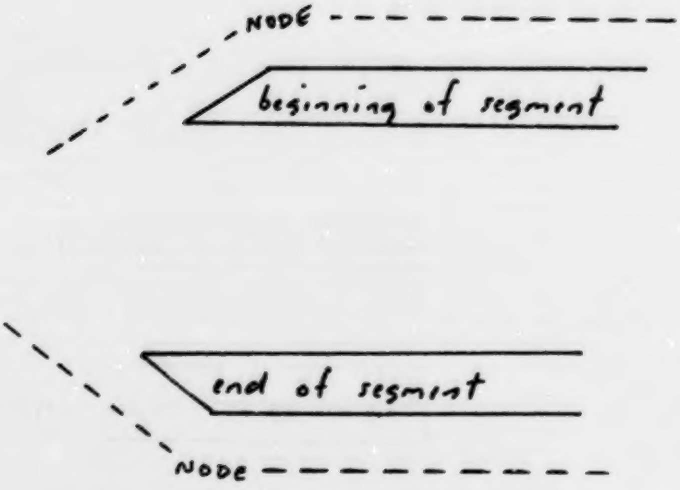
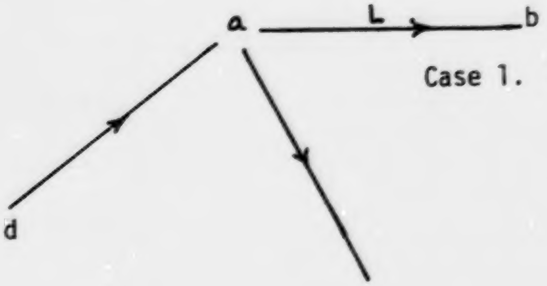
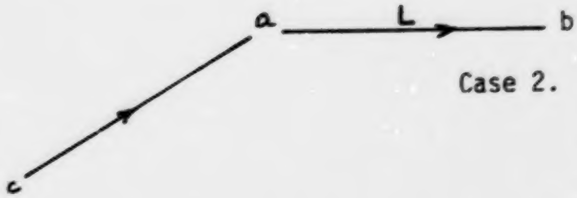

NODE
beginning of segment
end of segment
NODE

Fig. 8.47d. Configuration type "D"

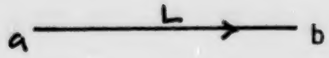
The type of configuration for a given link depends on the nature of the links entering and exiting the link. For the vehicle volume plot the following situations may occur. In this description $\longrightarrow$ represents a regular link, ----> represents a centroid connector or a turning link, $\rightsquigarrow$ represents a link with one node external to the plot area. $\xrightarrow{L}$ is link under consideration.



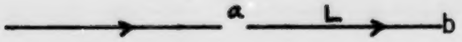
Case 1. At a, configuration 8.47d



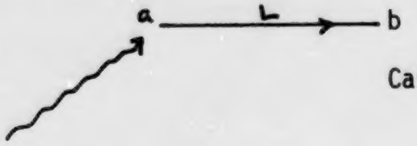
Case 2. At a, configuration 8.47c. If volume on c-a is zero, configuration is 8.47 d.



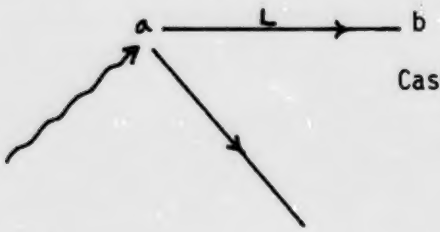
Case 3. At a, configuration 8.47a



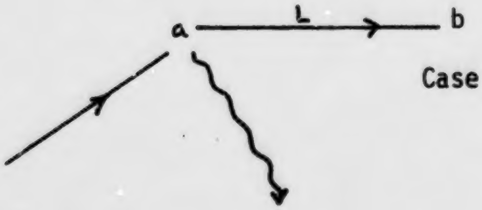
Case 4. At a, configuration 8.47b



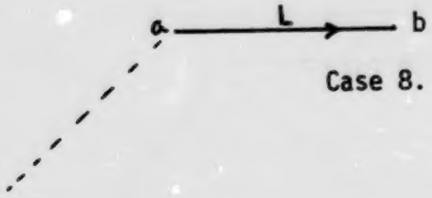
Case 5. At 2, configuration 8.47a



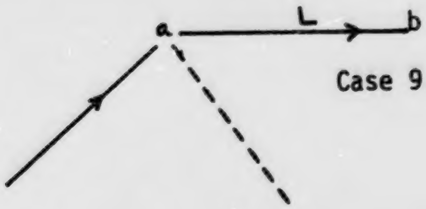
Case 6. At a, configuration 8.47d



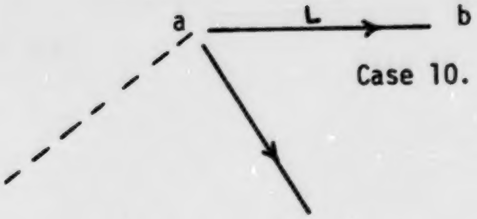
Case 7. At a, configuration 8.47d



Case 8. At a, configuration 8.47d

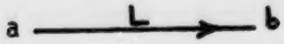


Case 9. At a, configuration 8.47d

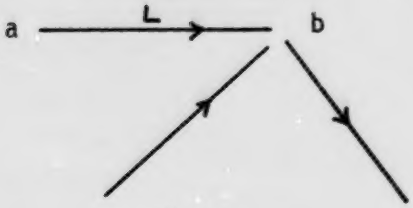


Case 10. At a, configuration 8.47d

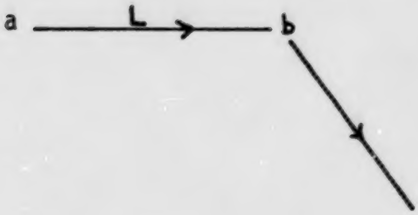
End Configurations



Case 1. At b, configuration 8.47a

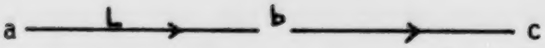


Case 2. At b, configuration 8.47d

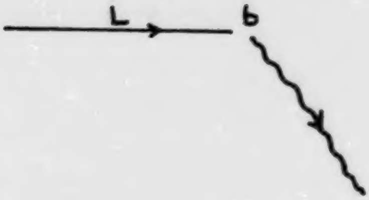


Case 3. At b, configuration 8.47d

If volume on b-c- is zero configuration 8.47d



Case 4. At b, configuration 8.47b



Case 5. At b, configuration 8.47a

OUTPUT:		
	File name:	GPQRAN
	Type :	Work file
	Symbol :	g_{pq}^{car}
	Status :	Always created
	Contents :	Matrix of auto trips between each origin and all destinations.
	Sequence :	Random, indexed by origin
	Use :	Initial value of demand in traffic assignment with variable demand.

PROCEDURE: (For option S1, the percentage of total trips made by auto is the same for all origins and destinations and is specified in a subcommand initially if the user specifies a value, otherwise the system default = 1.0 is used. Options S3 and S4 are handled in 6.10 AUTRIP). The program functions are summarized in tabular form following this description.

For each origin

- (Options S1, S2) Read in total equivalent vehicle trips from file GBPQ
- (Option S2) Read in percentage of auto trips from file MODEST
- (Options S1, S2) Calculate auto trips

$$g_{pq}^{car} = \bar{g}_{pq}^{veh} * m \qquad \text{Option S1}$$

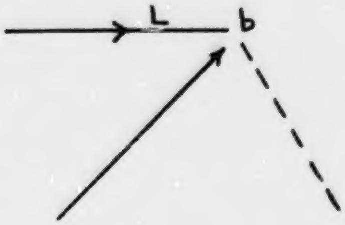
$$g_{pq}^{car} = \bar{g}_{pq}^{veh} * m_{pq} \qquad \text{Option S2}$$

- (Option S5) Read in estimate of number of auto trips from file MODEST
- Add vehicle adjustment

$$g_{pq}^{car} = g_{pq}^{car} + e_{pq}^{car} \qquad q = \text{all destinations}$$

(e_{pq}^{car} is assumed to be zero if ODEXSW in /APARM/ is FALSE i.e. subcommand parameter VEAD = NO).

- Write auto trips for origin as indexed record (by origin) on file GPQRAN.



Case 10. At b, configuration 8.47d

The blacked-in area representing the traffic volume has the structure as shown in Fig. 8.05a. The parameters completely determining this structure are (X_b, Y_b), ϕ_b, (X_e, Y_e), ϕ_e and LVOL. Routine AVLINE determines the type of configuration of the volume line for a given link, calculates $(X_b m\ Y_b)$, ϕ_b, (X_e, Y_e) abd ϕ_e, by accessing one of the routines CONFGA, CONFGB, CONFGC or CONFGD and completes the plot of the volume segment with calls to VOL and VOLNUM.

MODULE : 8.48 CNUM

PURPOSE : Plot volume number on centroid connector for vehicle volume plot.

INPUT : a) Data in COMMON blocks /NOEUD/, /GEOM/ and /NETSIZ/

XNO(K)	x-coordinate for node K.
YNO(K)	y-coordinate for node K.
WSYM	Quotient width/height for a plotted character using the system supplied plot routine SYMBOL[+].
NCENT	Number of zone centroids.

b) Subroutine parameters

NP1	Origin node of link.
NP2	Destination node of link.
NUM	Predicted link volume.
ALPHA	Angle of link to x-axis.

PROCEDURE :
1. If link length < 0.6" set IFLAG to 1; otherwise IFLAG=0.
2. Determine number of digits in predicted volume.
3. Determine location for plotting link volume. Volume is plotted near zone centroid, parallel to link median, displaced to the right with respect to the link direction.
4. If IFLAG=1, plot asterisk (link too short to contain number). Otherwise plot number.
5. Return

SYSTEM PLOT
ROUTINES: NUMBER[+], SYMBOL

[+] CDC system plot routine

MODULE : 8.61 TRNETP

PURPOSE : Transit network plot MAIN program

INPUT : Data is accessed through routines OPBAN, DINTRN and a BLOCK DATA subprogram.

a) Data from OPBAN

XINFIN	Large positive number
NNODES	Largest transit node number
NCENT	Number of zone centroids

b) Data from DINTRN (module 8.62)

ANGROT	Angle in counterclockwise direction between R and R' where R is the system of reference of the geocoded coordinates and R' is the system of reference for the plot.
NX1,NX2	Node numbers giving lower and upper bounds for x-axis after rotation through ANGROT.
NY1,NY2	Node numbers giving lower and upper bounds for y-axis after rotation through ANGROT
XNO,YNO	Vectors of x and y node coordinates

c) Data from BLOCK DATA (module 8.09)

YMAX	Maximum permissible plot height (inches).

PROCEDURE : In the following description, names in left hand column are subroutines which contain code for the section described to the right.

1.	Initialize OK to TRUE.
OPBAN	Initialize data bank. Set OK to FALSE if any errors encountered. If OK is true go to step 2; otherwise exit.

2. DINTRN Read in user supplied plot data (See 8.62 DINTRN).

3. LIMITE+ Request maximum permissible plot height.

4. Rotate axes through ANGROT. Translate coordinates so that minimum x and y values are zero. Initialize ICORR to 0.

5. Check that node with maximum y-coordinate will fit on plot. If not, print error message and exit.

6. RDCOR Read in one record from node correspondence table.
   NODE is transit node whose coordinate has been defined.
   NB is number of nodes corresponding to NODE.
   KOR contains set of nodes corresponding to NODE

7. If end of file, go to step 9. If NB=0, return to step 6. If NB>1 go to step 8. Otherwise set coordinates of node KOR(1) equal to those of NODE and define correspondence between the two nodes using ICORR:

   ```
   ICORR(NODE)   = KOR(1)
   ICORR(KOR(1)) = NODE
   ```

   Return to step 6.

8. Determine if one of the nodes in KOR has a correspondence. If affirmative set up correspondence with ICORR. Return to step 6.

9. ROUTE Determine bus lines using each transit link.

10. Set coordinates of zone centroids outside plot area to XINFIN.

11. Print link table. SORT arranges transit line numbers for each link in numerical sequence. A maximum of 16 line numbers is retained for each link.

+CDC system routine

12. TRPLOT Plot transit network

13. End.

MODULE : 8.62 DINTRN

PURPOSE : Input user supplied data for transit network plot.

INPUT : a) Input file

File name: PSPEC

Type : Direct entry input file

Status : Required

Contents : M1, M2 - external node numbers of nodes defining lower and upper boundaries along x-axis after rotation of plot axes.

N1, N2 - external node numbers of nodes defining lower and upper boundaries along y-axis after rotation of plot axes.

I1, I2, J1, J2 - external node numbers of nodes defining a rectangle internal to plot area to be omitted from plot. I1 and I2 define lower and upper bounds of rectangle along x-axis (after rotation); J1 and J2 define lower and upper bounds of rectangle along y-axis. If no segment is to be omitted, I1 is 0.

SCALE - desired scale of plot (ft./in.).

ANGROT - angle in counterclockwise direction between R and R' where R is the system of reference of the geocpded coordinates and R' is the system of reference for the plot.

ANGNOR - angle between x-axis and north, counterclockwise sense in system of reference R.

LIND - number of bus lines for which volumes are desired (0 if all lines desired).

LINENO - vector containing bus line numbers of lines to be included in plit. Not required if LIND=0.

Sequence : Sequential.

Source : User specified

b) Data in COMMON areas /NETSIZ/, /PLUNIT/ and /CONST/

NNODES	Largest transit node number
PSPEC	File number of input file PSPEC specified in BLOCK DATA (module 8.09)
INFINI	Large integer constant
XINFIN	Large real constant

OUTPUT : Data in COMMON areas /NOEUD/, /BOUND/ and /INTEX/

XNO(K)	x-coordinate of node K
YNO(K)	y-coordinate of node K
EXTN(K)	External node number of node K.
NX1,NX2 NY1,NY2	Internal node numbers of M1, M2, N1 and N2 respectively.
IX1,IX2 IY1,IY2	Internal node numbers of I1, I2, J1 and J2 respectively.

PROCEDURE : 1. Rewind file PSPEC

2. Read in data from PSPEC

3. Determine scale specifying number of coordinate units per mile (routine GTSCAL).

4. Access coordinates from data bank using routine GTNODE.

5. Determine internal node numbers of nodes defining boundaries using routine PTNODE.

6. Return

MODULE : 8.63 ROUTE

PURPOSE : Determine bus lines using each link

INPUT : a) Data in COMMON areas /LINEP/, /NETSIZ/, /BOUND/ and /NOEUD/

LIND	Number of bus lines included in plot.
LINENO	Bus line numbers included in plot.
NNODES	Largest transit node number
XNO(K)	x-coordinate, node K
YNO(K)	y-coordinate, node K
IX1 IX2	Node numbers (internal), defining lower and upper bounds along x-axis of rectangle, internal to plot area, to be omitted from plot.
IY1 IY2	Node numbers (internal) defining lower and upper bounds along y-axis of rectangle, internal to plot area, to be omitted from plot.

b) Data statement.

LINKTR Upper bound for number of transit links.

c) Input file

File name: LINES

Type : Data base

Symbol

Status : Required

Contents : LISTN Node sequence on the itinerary of a given bus line

NDIR(J): Direction for node LISTIN(J)

KNODE : Number of elements in LISTN and NDIR for given bus line

Sequence : Sequential

Source : User specified

5. TABTN Insert transit line links in link table. Return to step 2.

6. Store external line numbers in LINENO.

7. Return,

MODULE : 8.64 TABTN

PRUPOSE : Insert transit links for bus line LINE in link table.

INPUT : a) Data in COMMON blocks /NOEUD/, /LINK/, /WORKC/, /NETSIZ/ and /BOUND/

XNO, YNO	Vectors of x and y coordinates for plotted region.
NO, ND	Origin and destination node vectors of links accumulated in link table from bus lines 1,...,LINE-1.
BLINE(J,K)	Bus lines serving link K from the LINE-1 lines already analysed.(K=1,NLINK, J=1,4).
LISTN	Vector containing node sequence for line number LINE.
NDIR(J)	Direction indicator for node LISTN(J).
NSEG	Number of elements in vector LISTN.
NOLAB(K)	Location of first occurrence of node K in link table.
NLINK	Cumulative number of links in link table for lines 1,...,LINES-1.
IX1	Node number defining lower bound along x-axis of rectangle internal to plot area to be omitted from plot.

b) Subroutine parameters

LINE	Current bus line number.
XU	Plot range x-axis is 0"-XU".
YU	Plot range y-axis is 0"-YU".
XIL	Lower x-coordinate of rectangle to be omitted from plot.
XIU	Upper x-coordinate of rectangle to be omitted from plot.
YIL	Lower y-coordinate of rectangle to be omitted from plot.
YIU	Upper y-coordinate of rectangle to be omitted from plot.

OUTPUT : Data in COMMON blocks /LINK/ and /NETSIZ/

NO(J), ND(J)	Origin and destination nodes for links accumulated in link table for lines 1,...,LINE (J=1,NLINK).
NOLAB(K)	Location of first occurence of node K in link table.
BLINE(J,K)	Cumulative bus lines serving link K (J=1,4; K=1,...,NLINK)
NLINK	Cumulative number of links in link table.

PROCEDURE : In the following description, names in the left hand column are subroutines which contain code for the section described to the right.

1. Initialize K to 1.

2. Check if node LISTN(K) lies on bus line itinerary in forward search through LISTN. If affirmative go to step 4.

3. Increment K by 1. If K=NSEG begin backward search through LISTN at step 12. Othewise return to step 2.

4. Set I=K+1.

5. Check if node LISTN(I) lies on bus line itinerary in forward search through LISTN. If affirmative go to step 7.

6. Increment I by 1. If I=NSEG begin backward search through LISTN at step 12. Otherwise return to step 5.

7. Set NORIG=LISTN(K); NDEST=LISTN(I). If either node lies outside plot area begin search for next link on line at step 11. Otherwise proceed to next step.

8. Check if link NORIG-NDEST already in link table. If affirmative go to step 10. Otherwise determine location INS for link to be inserted so that links are in order of increasing origin node numbers.

9. INSERT Insert link in link table at location INS. SET BLINE(1,INS)=LINE. Renumber links following location INS. Go to step 11.

10. TRANV Append line number LINE to BLINE array for link number INS.

11. Set K=1. If K=NSEG begin backward search through LISTN at step 12. Otherwise return to step 4.

12. Repeat similar procedure for a backward search through LISTN.

13. Return.

MODULE : 8.65 INSERT

PURPOSE : Insert new link in link table, transit network plot.

INPUT : a) COMMON blocks /LINK/ and /NETSIZ/

NO, ND	Origin and destination node vectors of links accumulated in link table.
NOLAB(K)	Location of first occurence of node K in link table.
BLINE(I,J)	Bus lines serving link J (J=1,NLINK, I=1,4).
NLINK	Cumulative number of links in table.

b) Subroutine parameters

INS	Location in link table for the insertion of current link.
NORIG	Origin node of current link
NDEST	Destination node of current link
LINE	Line number of current bus line.

OUTPUT : COMMON blocks /LINK/ and /NETSIZ/

Updated values for NO, ND, NOLAB, BLINE and NLINK.

PROCEDURE : 1. If link table is empty or if INS>NLINK go to step 3.

2. Reorder links in link table following location INS. Redefine NOLAB.

3. Increment NLINK by 1. Set NO(INS)=NORIG, ND(INS)=NDEST, BLINE(1,INS)=LINE and if NOLAB(NORIG)=0, set NOLAB(NORIG)=INS.

4. Return.

MODULE : 8.66 TRANV

PURPOSE : Append current bus line to the list of lines using a link, transit network plot.

INPUT : a) Data in COMMON block /LINK/

BLINE(J,K) Bus lines serving link K (J=1,4)

b) Subroutine parameters

INS Location of link in table.

LINE Line number to be appended to BLINE.

OUTPUT : Data in COMMON block /LINK/

BLINE(J,INS) Updated list of line numbers serving link INS (J=1,4).

PROCEDURE : 1. Determine storage location for line number in BLINE.

2. Append line number

3. Return.

MODULE : 8.67 SORT

PURPOSE : Order line numbers serving a link in an ascending sequence.

INPUT : a) Data in COMMON blocks /LINK/ and /LINEP/

BLINE(J,I)	Contains line numbers serving link I (J=1,4). For each value of J, up to 4 numbers may be stored; the number abcdefgh implies line numbers ab, cd, ef and gh.
LINENO(K)	External representation of internal line number K.

b) Subroutine parameters

I	Link number of link

OUTPUT : Subroutine parameters

IVECT	Vector of external line numbers serving link.
NVECT	Number of elements in IVECT.

PROCEDURE : 1. Store line numbers serving link in JVECT.

2. Sort numbers in JVECT in ascending sequence; store in IVECT.

3. Return.

CF 8/5/78

MODULE : 8.68 TRPLOT

PURPOSE : Determine number of links exiting each node in plot area, access plot routines, transit network plot.

INPUT : a) Data in COMMON blocks /NOEUD/, /LINK/ and /NETSIZ/

XNO, YNO	Vectors of x and y node coordinates
ND(L)	Origin node of link L (L=1,NLINK)
NLINK	Number of links in plot area.
NNODES	Largest transit node number.

b) Subroutine parameters

XU	Plot range, x-axis is 0"-XU"
YU	Plot range, y-axis is 0"-YU"

OUTPUT : Data in COMMON blocks /NOEUD/, /WORKC/ and /LINK/

XNO	Node x-coordinate vector translated so that network plot is centered with respect to plot title.
NEXIT	Vector indicating number of links exiting each node in plot area.

PROCEDURE : In the following description, names in the left hand column are subroutines which contain the code for the section described to the right.

1.		Initial NEXIT to 0.
2.		Determine NEXIT(K), the number of links exiting node K in plot area.
3.	PTITLE	Plot title of plot (see module 8.01).
4.		Translate x-axis of node coordinates to ensure the title is centered with respect to the network.
5.	TNALGO	Plot network (module 8.69).
6.	ENDPLT[+]	Indicate end of plot file.

[+] CDC system plot routine

CF 8/5/78

MODULE : 8.69 TNALGO

PURPOSE : Plot links and line numbers for transit network.

INPUT : a) Data in COMMON blocks /LINK/, /WORKC/ and /NETSIZ/

NO, ND	Origin and destination node vectors for links in plot area.
NOLAB(K)	Location of first occurence of node K in link table.
NEXIT(K)	Number of links in plot area exiting node K.
NLINK	Number of links in link table for plot area.
NNODES	Largest transit node number (internal node numbers).

PROCEDURE : In the following description, names in the left hand column are subroutines which contain code for the section described to the right.

1. Initialize NPLOT to FALSE, LSTART to 1.

2. Find first unplotted link in link table starting at location LSTART. If all links have been plotted, go to step 12. Otherwise set LINK and LSTART equal to link number of first unplotted link.

3. TNNODE Plot origin node number NP1 of LINK if not already plotted. Set NPLOT(NP1) to TRUE.

4. FLECHE Plot link. If LINK is one-way plot arrow.

5. TNNODE Plot destination node number NP2 of link number LINK if not already plotted.

6. Increment NEXIT(NP1) by 100 (Number of links exiting NP1 which have been plotted is given by quotient NEXIT(NP1)/100).

7. If some links exiting NP1 remain to be plotted go to step 8. Otherwise remove NP1 from cumulative table NTABLE. This vector accumulates nodes encountered from starting point LSTART. When all links exiting a given node have been plotted, the node number is removed.

8.		Investigate NP2 as starting point for next link plot. Set NP1=NP2. If no links have been plotted from NP2, enter NP2 in NTABLE.
9.		Determine if all links from NP1 have been plotted. If negative go to step 11.
10.		Locate first node in NTABLE having an unplotted exiting link. If no links are found return to step 2. Otherwise set NP1 equal to node number.
11.		Find first unplotted link from NP1. Go to step 4.
12.	TNACES	Plot access links.
13.		Return.

MODULE : 8.70 FLECHE

PURPOSE : Plot link NP1-NP2, transit network plot.

INPUT : a) Data in COMMON block /CORR/

ICORR(K) Node number corresponding to node K if K is doubly defined; 0 otherwise.

b) Subroutine parameters

LINK	Link number of link being plotted.
NP1	Origin node of link.
NP2	Destination node of link.

PROCEDURE : 1. Initialize IPEN to 2 and IARR to 1.

2. Check if link median has already been drawn; if affirmative set IPEN=3. Check if link is two-way; if affirmative set IARR to 0.

3. Using routine LINKTN (module (8.73)) plot link median if IPEN=2. Plot bus line numbers, and if IARR=1, plot arrow on link median indicating direction of one-way link.

4. Return.

MODULE : 8.71 CHECTN

PURPOSE : Check if given link is two-way and if link has already been plotted (transit network plot).

INPUT : a) Data in COMMON blocks /LINK/, /WORKC/ and /NETSIZ/

NO(I)	Origin node of link number I.
ND(I)	Destination node of link number I.
NOLAB(K)	Location of first occurrence of node K in link table.
NEXIT(K)	Equal to 100x (number of links exiting node K which have been plotted) + number of links exiting node K.
NLINK	Number of links in link table.

b) Subroutine parameters

NP1	Origin node of link.
NP2	Destination node of link.

OUTPUT : Subroutine parameters

TWAY	TRUE if link is two-way.
APLOT	TRUE if link has already been plotted.

PROCEDURE :

1. Initialize APLOT and TWAY to FALSE
2. If link NP2-NP1 is not in link table RETURN; otherwise go to step 3.
3. Check if NP2-NP1 has been plotted; if affirmative set APLOT to TRUE.
4. Return.

CF 14/11/78

MODULE : 8.72 TNNODE

PURPOSE : Plot node number, transit networt plot.

INPUT : a) Data in COMMON blocks /NOEUD/, /INTEX/, /GEOM/, and /CORR/

XNO(NP)	x-coordinate for node NP.
YNO(NP)	y-coordinate for node NP.
EXTN(NP)	external node number of node NP.
ANGROT	Angle, in counterclockwise sense, through which plot axes are rotated with respect to axes of geo-coded coordinates.
CT, ST	Cosine and sine of ANGROT
ICORR	Defines correspondence between transit and vehicle nodes. ICORR(NP)=0 if NP has no corresponding nodes; otherwise ICORR(NP) is node number corresponding to NP.

PROCEDURE :
1. Plot circle, radius .04" at node location using routine SYMBOL[+].
2. Determine location for plotting node number.
3. Plot number using routine NUMBER[+].
4. Return.

SYSTEM PLOT
ROUTINES: SYMBOL and NUMBER

[+] CDC system plot routine

CF 10/5/78

MODULE : 8.73 LINKTN

PURPOSE : Plot link median and bus line numbers, transit network plot.

INPUT : a) Data in COMMON blocks /NOEUD/, /LINK/ and /CORR/

XNO, YNO	Vectors of x and y coordinates for all nodes.
BLINE(J,LINK)	Contains bus line numbers serving link LINK(J=1,4).
ICORR(K)	Defines correspondence between transit and vehicle nodes. ICORR(K)=0 if node K has no corresponding nodes; otherwise ICORR(K) denotes node number corresponding to K.

b) Subroutine parameters

NP1	Origin node for link.
NP2	Destination node for link.
IARR	0 for two-way link; 1 for one-way link.
IPEN	2 if link median is plotted; 3 if link median is not plotted. 4 if link is a centroid connector.
LINK	Link number of current link

PROCEDURE : 1. Calculate beginning and end points for link median.

2. If transit line numbers only are to be plotted go to step 6. If link is centroid connector go to step 3. Otherwise plot link median. Go to step 4.
3. Join nodes NP1 - NP2 with dashed line.
4. If link is one-way, plot arrow on median using routine SYMBOL[+].
5. If link is centroid connector RETURN.

[+] CDC system plot routine

6. Plot transit numbers parallel to link median

7. Return.

SYSTEM PLOT
ROUTINE: SYMBOL, PLOT, NUMBER.

MODULE : 8.74 TNACES

PURPOSE : Plot centroid connectors

INPUT : a) Data in COMMON blocks /NOEUD/, WORKC/, /CORR/, /NETSIZ/ and /CONST/

XNO, YNO	Vectors of x and y coordinates for nodes in plot area.
NPLOT	Vector indicating if node number has been plotted. NPLOT(K) is TRUE if node K has been plotted; FALSE otherwise.
ICORR	Defines correspondence between road and transit nodes; ICORR(K) is node number corresponding to node K or 0 if node K is not in correspondence table.
NCENT	Number of zone centroids in network.
NNODES	Largest transit node number.
XINFIN	Large real number.

b) Input file: ACCESS-EGRESS

Type : Data base

Symbol :

Status : Required

Contents : Transit access and egress links

Sequence : Sequential

Use : Furnish access and egress links for transit network plot.

PROCEDURE : In the following description, names in the left hand column are subroutines which contain code for the section described to the right.

1.	PSACCL	Rewind file containing access links (ACCESS).
2.	PSEGRL	Rewind file containing egress links (EGRESS).
3.		Initialize NOLAB and NLINK to 0.
4.	RDACCL	Read in access link record from ACCESS. If end of file, go to step 6.

5.	If destination node of link has not been plotted or if centroid node number outside plot area, return to step 4. Otherwise, add link to link table and increment NLINK by 1. Return to step 4.
6. RDEGRL	Read in egress link record from EGRESS. If end of file encountered go to step 8.
7.	If origin node of link has not been plotted or if centroid node is outside plot area, return to step 6. Otherwise add link to link table and increment NLINK by 1. Return to step 6.
8.	Plot access links.
9.	Plot egress links
10.	Return.

MODULE : 8.81 TRVOLP

PURPOSE : Transit volume plot MAIN program

INPUT : Data is accessed through routines OPBAN, DINTRV and a BLOCK DATA subprogram.

a) Data from OPBAN

XINFIN	Large positive constant
NNODES	Largest transit node number
NCENT	Number of zone centroids

b) Data from DINTRV (module 8.82)

VMAX	Upper bound for transit link volumes
ANGROT	Angle in counterclockwise direction between R and R' where R is the system of reference of the geocoded coordinates and R' is the system of reference for the plot.
NX1, NX2	Node numbers giving lower and upper bounds along x-axis of plot after rotation through ANGROT
NY1, NY2	Node numbers giving lower and upper bounds along y-axis of plot after rotation through ANGROT
XNO(K)	x-coordinate of node K
YNO(K)	y-coordinate of node K

c) Data from BLOCK DATA subprogram (module 8.09)

DELTA, DELTA1	Parameters determining the points of intersection of two volume lines for two connected links
YMAX	Maximum permissible plot height (inches)

PROCEDURE : In the following description, names in left hand column are subroutines which contain code for the section described to the right.

1. Initialise OK to TRUE.

OPBAN Initialise data bank. Set OK to FALSE if any errors encountered.

2. DINTRV Read in user supplied plot data. (see 8.82 DINTRV).

3. LIMITE* Request maximum permissible plot height.

4. Fix plot parameters. EPS and R determine distance of volume lines from node number. NV designates the transit volume represented by 0.01".

5. Rotate axes through ANGROT. Translate coordinates so that minimum x and y values are zero. Initialise ICORR(K) and JCORR(K), (K = 1, NNODES), to zero.

6. Check that node with maximum y-coordinate will fit on plot. If not, print error message and exit.

7. RDCOR Read in one record from node correspondence table. NODE is transit node whose coordinate has been defined. NB is number of nodes corresponding to NODE. KOR contains set of nodes corresponding to NODE.

8. If end of file, go to step 10. If NB = 0 return to step 7. If NB > 1 go to step 9. Otherwise set coordinates of node KOR(1) equal to those of NODE and define correspondence between the two nodes using ICORR:

ICORR(NODE) = KOR(1)
ICORR(KOR(1)) = NODE

Return to step 7.

---

* CDC system routine

9.		Determine if one of the nodes in KOR has a correspondence. If affirmative set up correspondence with ICORR. Return to step 7.
10.		Set coordinate values for nodes outside plot area to XINFIN.
11.	VOLACC	Accumulate transit link volumes for bus lines in plot area.
12.		Print transit link table and link volumes.
13.	PLTRVL	Plot network and link volumes
14.		End.

MODULE : 8.82 DINTRV

PURPOSE : Input user supplied data for transit volume plot.

INPUT : a) Input file

File name : PSPEC

Type : Direct entry input file

Status : Required

Contents : M1, M2 - external node numbers of nodes defining lower and upper boundaries along x-axis after rotation through ANGROT

N1, N2 - external node numbers of nodes defining lower and upper boundaries along y-axis after rotation through ANGROT

I1, I2, J1, J2 - external node numbers of nodes defining a rectangle internal to plot area, to be omitted from plot. I1 and I2 define lower and upper bounds of rectangle along x-axis (after rotation); J1 and J2 define lower and upper bounds of rectangle along y-axis. If no segment is to be omitted, I1 is 0.

SCALE - desired scale of plot (ft./in.)

ANGROT - angle in counterclockwise direction between R and R' where R is the system of reference of the geo-coded coordinates and R' is the system of reference for the plot.

ANGNOR - angle between x-axis and north, counterclockwise sense in system of reference R.

VMAX - upper limit for link volume.

LIND - number of bus lines for which volumes are desired. (0 if all lines desired)

LINENO - vector containing bus line numbers of links to be included in plot. Not required if LIND = 0.

Sequence : Sequential

Source : User specified

b) Data in COMMON areas /NETSIZ/, /PLUNIT/ and /CONST/

NNODES	Largest transit node number
PSPEC	File number of input file PSPEC specified in BLOCK DATA (module 8.09).
XINFIN	Large real positive constant.
INFINI	Large positive constant (integer)

OUTPUT : Data in COMMON areas /NOEUD/, /BOUND/ and /INTEX/.

XNO(K)	x-coordinate of node K
YNO(K)	y-coordinate of node K
EXTN(K)	External node number of node K
NX1, NX2 NY1, NY2	Internal node numbers of M1, M2, N1 and N2 respectively.
IX1, IX2 IY1, IY2	Internal node numbers of I1, I2, J1 and J2 respectively.

PROCEDURE :

1. Rewind file PSPEC
2. Read in data from PSPEC
3. Determine scale specifying number of coordinate units per mile (routine GTSCAL)
4. Access coordinates from data bank using routine GTNODE

5. Determine internal node numbers of nodes defining boundaries using routine PTNODE.
6. Return.

MODULE : 8.83 VOLACC

PURPOSE : Construct transit link table; accumulate link volumes transit volume plot.

INPUT : a) Data in COMMON blocks /LINEP/, /NETSIZ/ and /BOUND/.

LIND	Number of bus lines included in plot
LINENO	Bus line numbers included in plot
NNODES	Largest transit node number
IX1,IX2 IY1,IY2	Internal node numbers of nodes defining region internal to plot area to be omitted from plot. Link I-J is omitted if XNO(IX1)<XNO(K)<XNO(IX2) and YNO(IY1)<YNO(K)<YNO(1Y2) for K = I or J

b) Data statement

LINKTR	Upper bound for number of transit links

c) Input files

1. File name : TNET

Type : Intermediate file

Symbol :

Status : Required

Contents : Vectors representing transit network structure.

Variables required by VOLACC are:

- NLINE : Number of transit lines
- NOLIST(L) : Pointer for the beginning of the itinerary of line L in vector LISTN (L=1,NLINE)
- NONLIN(L) : Pointer for the end of the itinerary of line L in vector LISTN (L=1,NLINE)

Sequence : Sequential

Use : Furnishes parameters for reading transit volumes.

2. File name : TFLOW

Type : Intermediate file

Symbol :

Status : Always created

Contents : Transit network predicted headways, travel times and passenger volumes.

Sequence : Sequential

Use : Furnishes transit link volumes for plot

3. File name : LINES

Type : Data base

Symbol :

Status : Required

Contents : Node sequence for each transit line

LISTN(J) : Node sequence on the itinerary of a given bus line.

NDIR(J) : Direction indicator for node LISTN(J)

KNODE : Number of elements in LISTN and NDIR for given bus line

Sequence : Sequential

Use : Furnish transit line description

d) Subroutine parameters

XU x-range of plot is 0"-XU"

YU y-range of plot is 0"-YU"

OUTPUT : Data in COMMON blocks /NETSIZ/ and /LINK/

NLINK : Number of links generated in plot area

NO(L) : Origin node number of link L (L=1,NLINK).

ND(L) : Destination node number of link L (L=1,NLINK)

NOLAB(K) : Location of first occurence of node K in link table (arranged in order of increasing origin node numbers)

LVOL(L) : Transit volume on link L (L=1,NLINK)

PROCEDURE : In the following description, names in left hand column are subroutines which contain code for the section described to the right.

1. Rewind TNET

2. Read in NOLIST and NONLIN from TNET

3. Initialise NOLAB, LVOL, NLINK and L to 0.

4. RDLINE Read in line description for line L+1 from file INES. Store in LISTN and NDIR.

5. If end of file is reached go to step 9. Otherwise increment L by 1 and read in transit volumes for line from TFLOW. Store volumes in DY and DZ.

6. Set number of records in line description to NSEG

7. If line is to be included in plot go to step 8. Otherwise return to 4.

8. ORDER Insert transit line links in link table and accumulate link volumes for links in plot area. Return to step 4.

9. Rewind TNET and TFLOW.

10. Return.

MODULE : 8.84 ORDER

PURPOSE : Insert transit link in plot area for bus line L in link-table, transit volume plot.

INPUT : a) Data in COMMON blocks /NOEUD/, /LINK/, /WORKC/, /LINVL/, /NETSIZ/ and /CONST/.

XNO,YNO	Vectors of x and y coordinates for plotted region.
NO,ND	Origin and destination node vectors of links accumulated in link table from bus lines 1,...,L-1.
NOLAB(K)	Location of first occurence of node K in link table.
LVOL	Accumulated link volume vector for lines 1,...,L-1.
LISTN	Sequence of node numbers for line L describing line L.
NDIR(J)	Direction indicator for node LISTN(J).
DY(K)	Transit volumes for line L for link LISTN(K-1)-LISTN(K)
DZ(K)	Transit volumes for line L for link LISTN(K)-LISTN(K-1).
NLINK	Cumulative number of links in link table for lines 1,...,L-1.
NSEG	Number of elements in vector LISTN.
XINFIN	Large positive constant

b) Subroutine parameters:

XU	x-range of plot is 0"-XU".
YU	y-range of plot is 0"-YU".
XIL,XIU YIL,YIU	Links with one or two nodes internal to region defined by XIL<X<XIU, YIL<Y<YIU, are omitted from plot

OUTPUT : Data in COMMON blocks /LINK/ and /NETSIZ/

NO(J),ND(J)	Origin and destination nodes of links accumulated in link table for lines 1,...L. (J=1,NLINK)

NOLAB(K)	Location of first occurence of node K in link table (Links arranged in order of increasing origin node number).
LVOL(J)	Cumulative volume, link J, for lines 1,...,L.
NLINK	Cumulative number of links in link table for lines 1,...,L.

PROCEDURE : In the following description, names in the left hand column are subroutines which contain code for the section described to the right.

1.	Initialise K to 1.
2.	Check if node LISTN(K) lies on bus line itinerary in forward search through LISTN. If affirmative, go to step 4.
3.	Increment K by 1. If K=NSEG begin backward search through LISTN at step 13. Otherwise return to step 2.
4.	Set I=K+1.
5.	Check if node LISTN(I) lies on bus line in forward search through LISTN. If affirmative go to step 7.
6.	Increment I by 1. If I=NSEG begin backward search through LISTN at step 13. Otherwise return to step 5.
7.	Set NORIG=LISTN(K); NDEST=LISTN(I). If both nodes lie outside plot area, begin search for next link on line at step 12. Otherwise go to next step.
8. TRANFR	If link multiply defined, use lower node numbers.
9.	Check if link NORIG-NDEST already in table. If affirmative go to step 11. Otherwise determine location INS for link to be inserted.
10. ACC1	Insert link in table at location INS. Set LVOL(INS) to DY(I). Renumber following links. Go to step 12.

11. ACC2 Add volume DY(I) to LVOL(INS).

12. Set K=I. If K=NSEG begin backward search through LISTN at step 13. Otherwise return to step 4.

13. Repeat similar procedure for a backward search through LISTN.

14. Return.

MODULE : 8.85 ACC1

PURPOSE : Insert link in link table, transit volume plot.

INPUT : a) COMMON blocks /LINK/ and /NETSIZ/

NO(J),ND(J)	Origin and destination node numbers of link J.
NOLAB(K)	Location of first occurence of node K in link table.
LVOL(J)	Cumulative volume for link J (J=1,NLINK).
NLINK	Cumulative number of links in table.

b) Subroutine parameters

INS	Location in link table for the insertion of the current link
NORIG	Origin node of current link
NDEST	Destination node of current link
VOL	Transit volume on link from current bus line.

OUTPUT : COMMON blocks /LINK/ and /NETSIZ/

Updated values for NO, ND, NOLAB, LVOL and NLINK.

PROCEDURE :

1. If link table is empty or if INS>NLINK go to step 3.
2. Reorder links in link table following location INS. Redefine NOLAB
3. Increment NLINK by 1. Set NO(INS)=NORIG, ND(INS)=NDEST, LVOL(INS)=VOL and if NOLAB(NORIG)=0, set NOLAB(NORIG)=INS.
4. Return.

MODULE : 8.86 ACC2

PURPOSE : Increment volume on link number INS by specified amount, transit volume plot.

INPUT : a) Data in COMMON block /LINK/

LVOL(INS) Cumulative link volume, link INS.

b) Subroutine parameters

INS Location of link in link table

VOL Volume to be added to LVOL(INS)

OUTPUT : Data in COMMON block /LINK/

LVOL(INS) Updated volume for link INS

PROCEDURE : 1. Add VOL to LVOL(INS)

2. Return.

MODULE : 8.87 TRANFR

PURPOSE : If link is multiply defined, use definition with lowest node numbers (transit volume plot routine).

INPUT : a) Data in COMMON block /CORR/

ICORR Vector giving correspondence for nodes which are assigned two node numbers.

b) Subroutine parameters

NORIG Origin node number of link

NODEST Destination node number of link.

OUTPUT : Subroutine parameters

NORIG New origin node number of link

NDEST New destination node number of link.

PROCEDURE : 1. If NORIG is unique node number for origin node of link, go to step 4.

2. If NORIG $<$ ICORR(NORIG) go to step 4.

3. Set NORIG=ICORR(NORIG).

4. If NDEST is unique node number for destination node of link, RETURN.

5. If NDEST $<$ ICORR(NDEST), RETURN.

6. Set NDEST=ICORR(NDEST).

7. Return.

MODULE : 8.90 PLTRVL

PURPOSE : Determine number of links exiting each node in plot area, access plot routine.

INPUT : a) Data in COMMON blocks /NOEUD/, /LINK/ and /NETSIZ/

XNO(K)	x-coordinate, node K.
YNO(K)	y-coordinate, node K.
NO	Origin node vector for links in link table.
ND	Destination node vector for links in link table.
NOLAB(K)	Vector giving location of first occurence of origin node K in link table.
LVOL	Vectors containing link volumes.
NNODES	Largest transit node number.
NLINK	Number of links in link table.

b) Subroutine parameters

XU	Plot range, x-axis is 0"-XU".
YU	Plot range, y-axis in 0"-YU".

OUTPUT : Data in COMMON blocks /NOEUD/, /WORKC/ and /LINK/

XNO	Node x-coordinates translated so that network plot is centered with respect to plot title.
NEXIT	Vector indicating number of links exiting each node in plot area.
LVOL(L)	If only one node of link L is in plot area, LVOL(L) = -1.

PROCEDURE : In the following description, names in the left hand column are subroutines which contain the code for the section described to the right.

1.		Initialize NEXIT to 0.
2.		Determine NEXIT(K), the number of links exiting node K in plot area. In one node of link L lies outside plot area set LVOL(L) = -1.
3.	PTITLE	Plot title of plot (see module 8.01).
4.		Translate x-axis so that title is centred on plot.
5.	HEADNG	Plot scale and direction indicator (see module 8.02).
6.	BAR	Plot volume scale for plot (see module 8.07).
7.	TVALGO	Plot network link volumes (see module 8.91).
8.	ENDPLT[+]	Indicate end of plot file.
9.		Return.

[+] CDC system plot routine

MODULE : 8.91 TVALGO

PURPOSE : Plot network and link volumes, transit network.

INPUT : Data in COMMON blocks /LINK/, /WORKC/ and /NETSIZ/.

NO,ND	Origin and destination node vectors for links in plot area.
NOLAB(K)	Location of first occurence of node K in link table.
LVOL	Link volume vector
NEXIT(K)	Number of links in plot area exiting node K.
NLINK	Number of links in link table for plot area.
NNODES	Largest transit node number.

PROCEDURE : In the following description, names in left hand column are subroutines which contain code for the section described to the right.

1.		Initialize NPLOT to FALSE, LSTART to 1.
2.		Find first unplotted link in link table starting at location LSTART. If all links have been plotted go to step 13. Otherwise set LSTART and LINK equal to link number of first unplotted link.
3.	TVNODE	Plot origin node number NP1 of LINK if not already plotted. Set NPLOT(NP1) to TRUE.
4.	CENLIN	Plot link median of link number LINK if not already plotted.
5.	TVNODE	Plot destination node NP2 of link number LINK if not already plotted. Set NPLOT(NP2) to TRUE.
6.	TVLINE	Plot volume lines and volume number for link number LINK.
7.		Increment NEXIT(NP1) by 100 (Number of links exiting NP1 which have been plotted is given by quotient NEXIT(NP1)/100).
8.		If some links exiting NP1 remain to be plotted go to step 9. Otherwise remove NP1 from cumulative table

NTABLE. This vector accumulates nodes encountered from starting point LSTART. When all links exiting a given node have been plotted, the node number is removed.

9. Investigate NP2 as starting point for next link plot. Set NP1=NP2. If no links have been plotted from NP2, add NP2 to NTABLE.

10. Determine if all links from NP1 have been plotted. If negative, go to step 12.

11. Locate first node in NTABLE having an unplotted exiting link. If no links are found, return to step 2. Otherwise, set NP1 equal to node number.

12. Find first unplotted link from NP1. If unplotted link traverses plot boundary (LVOL = -1) increment NEXIT(NP1) by 100 and continue search. If no exiting link found return to step 11. Otherwise set LINK equal to link number and return to step 4.

13. TVACES Plot access links.

14. RETURN.

MODULE : 8.92 CENLIN

PURPOSE : Plot link median, transit volume plot.

INPUT : a) Data in COMMON block /NOEUD/

XNO(K)	x-coordinate, node K
YNO(K)	y-coordinate, node K

b) Subroutine parameters

NP1	Origin node of link
NP2	Destination node of link.

PROCEDURE : In the following description, names in left hand column are subroutines which contain code for the section described to the right.

1.	CHECTV	APLOT is TRUE if median line has already been drawn. (module 8.93)
2.		If APLOT is TRUE, RETURN. Otherwise, set X1, Y1 and X2, Y2 equal to x and y coordinates of NP1 and NP2 respectively.
3.	DPARAM	Determine beginning and end point of median line. Also length and inclination of line to x-axis (module 8.10).
4.	DASH	Draw dashed line along median (module 8.11).
5.		Return.

MODULE : 8.93 CHECTV

PURPOSE : Check if median line for link has been drawn, transit volume plot.

INPUT : a) Data in COMMON blocks /LINK/, /WORKC/, and /NETSIZ/.

NO, ND	Origin and destination node vectors for links in plot area.
NOLAB(K)	Location of first occurence of node K in link table.
NEXIT(K)	NEXIT(K)/100 gives number of links exiting node K that have been plotted.
NLINK	Number of links in plot area.

b) Subroutine parameters

NP1	Origin node of link
NP2	Destination node of link

OUTPUT : Subroutine parameter

APLOT	TRUE if median line of link NP1-NP2 has been plotted, FALSE otherwise.

PROCEDURE : 1. Initialize APLOT to FALSE.

2. Determine if volume lines for link NP2-NP1 have been plotted. If affirmative set APLOT to TRUE.

MODULE : 8.94 TVNODE

PURPOSE : Plot link node number transit volume plot.

INPUT : a) Data in COMMON blocks /NOEUD/, /INTEX/ and /GEOM/

XNO(K)	x - coordinate, node K.
YNO(K)	y - coordinate, node K.
EXTN	Vector of external node numbers.
ANGROT	Angle through which plot is rotated with respect to geocoded coordinates.

b) Subroutine parameters

NP Internal node number of node to be plotted.

PROCEDURE : 1. Determine external node number NE of node.

2. Plot node number NE

SYSTEM PLOT ROUTINES : NUMBER[+]

[+] CDC system plot routine.

MODULE : 8.95 TVACES

PURPOSE : Plot access and egress links, transit volume plot.

INPUT : a) Data in COMMON blocks /NOEUD/, /WORKC/, /BOUND/ and /NETSIZ/.

XNO(K)	x-coordinate, node K.
YNO(K)	y-coordinate, node K.
NPLOT(K)	NPLOT(K) is TRUE if node K has been plotted; FALSE otherwise.
NX2, NY2	Nodes defining range of plot along x and y axes respectively.
NNODES	Largest transit node number.
NCENT	Number of node centroids in transit network.

b) Input file: ACCESS-EGRESS

Type : Data base.

Symbol :

Status : Required.

Contents : Transit access and egress links.

Sequence : Sequential.

Use : Furnish access and egress links for plot

PROCEDURE : In the following description, names in the left hand column are subroutines which contain code for the section described to the right.

1. PSACCL Rewind file containing access link (ACCESS).

2. PSEGRL Rewind file containing egress links (EGRESS).

3. Initialize NOLAB and NLINK to 0.

4. RDACCL Read in access link record from ACCESS. If end of file,go to step 6.

5. If destination node of link has not been plotted or if

centroid node outside plot area, return to step 4. Otherwise add link to link table and increment NLINK by 1. Return to step 4.

6. RDEGRL	Read in egress link record from EGRESS. If end of file encountered go to step 8.
7.	If origin node of link has not been plotted or if centroid node is outside plot area, return to step 6. Otherwise add link to link table and increment NLINK by 1. Return to step 6.
8.	Plot access links.
9.	Plot egress links
10.	Return.

MODULE : 8.96 TVLINE

PURPOSE : Determine structure of link volume segment transit volume plot, for a given link; access volume plot routines.

INPUT a) Data in COMMON blocks /NOEUD/, /PARAM/, /LINK/ and /NETSIZ/

XNO,YNO	Vectors of x and y coordinates for each node in plot area.
NV	Volume represented by 0.01".
NO, ND	Origin and destination node vectors for links in plot area.
NOLAB	Vector giving location of first occurence of each origin node in link table.
LVOL	Link volume vector.
NLINK	Number of links in plot area.

b) Subroutine parameters

NP1, NP2	Origin and destination nodes of link.
LINK	Link number of link where volume lines are being plotted.

PROCEDURE : The beginning and end sections of a link volume segment may each have one of four different configurations as shown by Figs. 8.96a - 8.96d

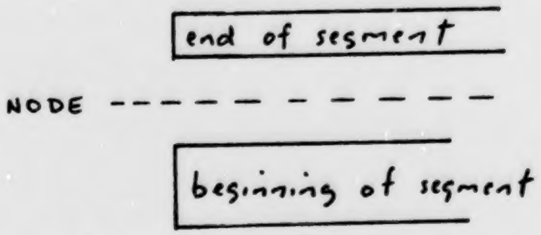


Fig. 8.96a Configuration type "A"

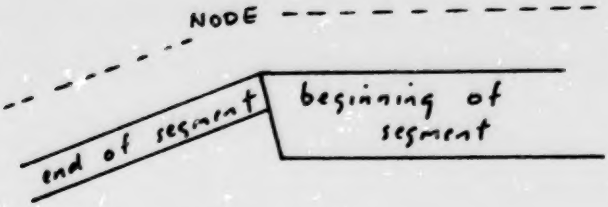

NODE
end of segment
beginning of segment

Fig. 8.96b Configuration type "B"

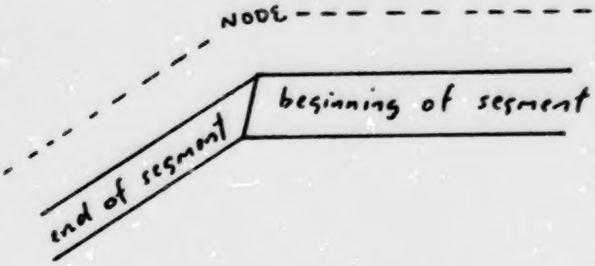

NODE
end of segment
beginning of segment

Fig. 8.96c Configuration type "C"

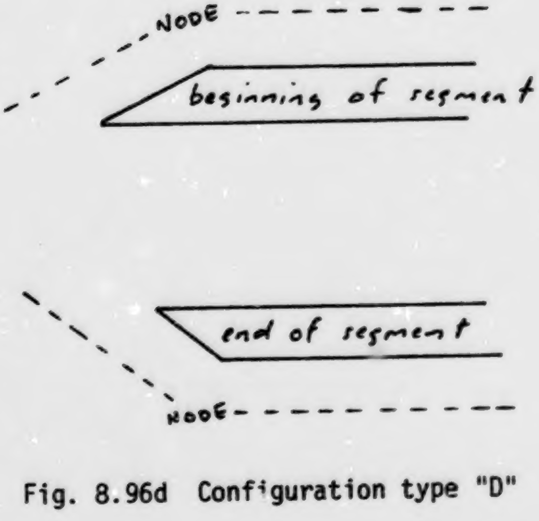
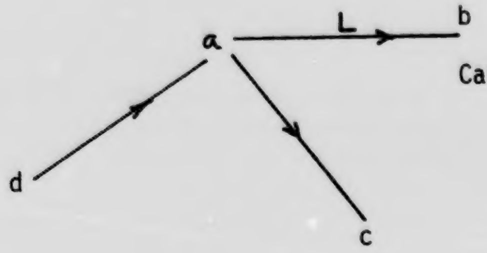

NODE
beginning of segment
end of segment
NODE

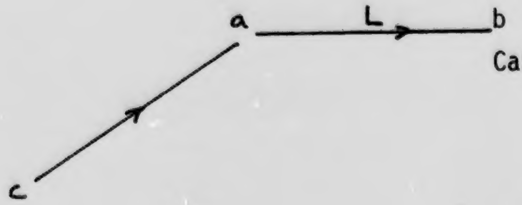
Fig. 8.96d Configuration type "D"

The type of configuration for a given link depends on the types of links entering and exiting the link. For the transit volume plot the following situations may occur. In this description ⟶ represents a regular link, ⇝ represents a link with one node external to plot area. $\xrightarrow{L}$ is the link under consideration.

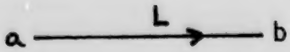
Beginning configurations.



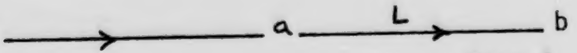
Case 1. At a, configuration 8.96d



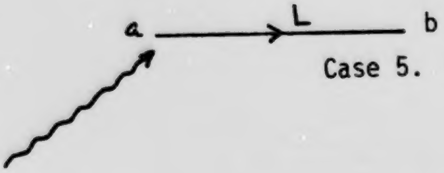
Case 2. At a, configuration 8.96c
If volume on c-a is zero, comfiguration is 8.96d



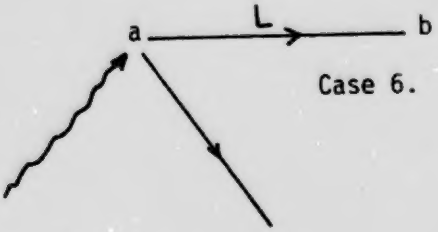
Case 3. At a, configuration 8.96a



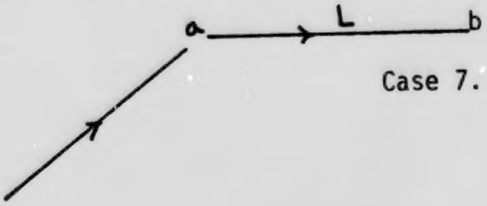
Case 4. At a, configuration 8.96b



Case 5. At a, configuration 8.96a

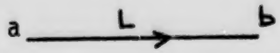


Case 6. At a, configuration 8.96d

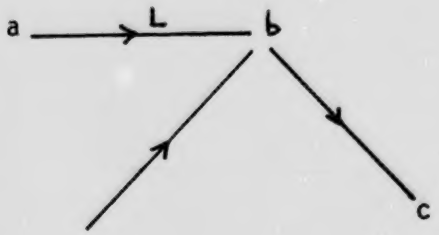


Case 7. At a, configuration 8.96d

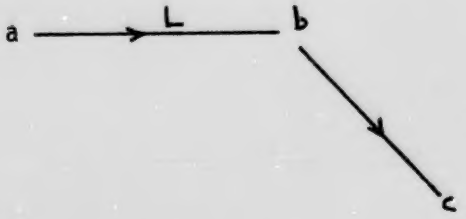
End configurations



Case 1. At b configuration 8.96a

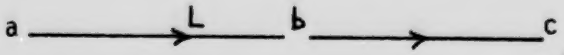


Case 2. At b confiburation 8.96d

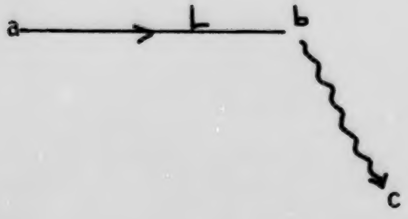


Case 3. At b, configuration 8.96c

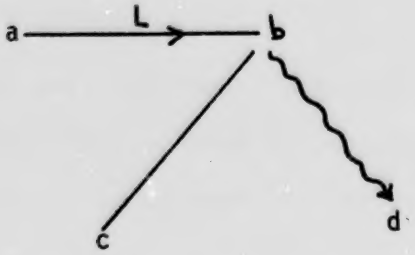
If volume on b-c is zero,
configuration 8.96d



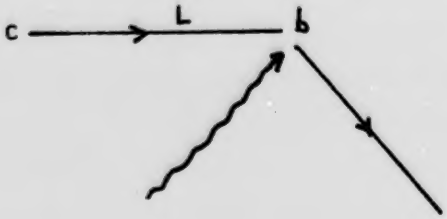
Case 4. At b, configuration 8.96b



Case 5. At b, configuration 8.96a



Case 6. At b configuration 8.96d



Case 7. At b configuration 8.96d

The blacked-in area representing the traffic volume has the structure as shown in Fig. 8.05a. The parameters completely determining this structure are (X_b, Y_b), ϕ_b, (X_e, Y_e) ϕ_e and LVOL. Routine TVLINE determines the type of configuration of the volume line for a given link, calculates (X_b, Y_b), ϕ_b, (X_e, Y_e) and ϕ_e, by accessing one of the routines CONFGA, CONFGB, CONFGC, or CONFGD and completes the plot of the volume segment with calls to VOL and VOLNUM.

Appendix C: File descriptions

This appendix contains descriptions of all the files used in the EMME system. The descriptions are arranged in order by file number. The file number corresponding to a particular file name can be determined from the cross reference index in Appendix D.

The terminology used in describing the files is as follows:

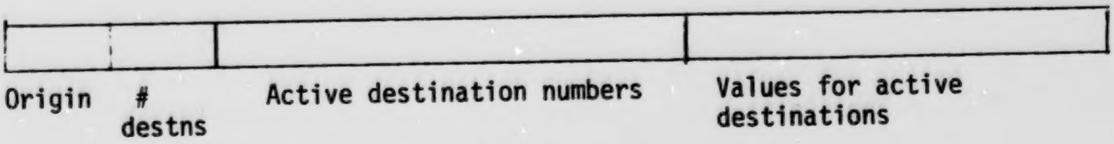
File name : Name of file, corresponds to name in COMMON UNIT
File number : Number assigned according to scheme described in paragraph 2.2
Contents : Self-explanatory
Type : Type as described for flowcharting standards in figure 2-1
Symbol : Algebraic symbol used to refer to contents
Created by : Module(s) which create file
Input to : Modules which read the file
File format : Self-explanatory
Record format : Self-explanatory
Accessed by : For data bank files only. Description of subroutine calls used to access the file.

Variables such as NLINE, NCENT etc used in describing the files correspond to the variables defining the size of the current network as defined for COMMON/NETSIZ/ in Appendix A.

Certain files are written in 'packed OD matrix format'. This is as follows.

Packed OD matrix file format

This format packs the non zero value V_{pq} contained in a matrix of rows (origins p) and columns (destinations q). The information for 1 origin (a row) is contained in one record of the following format.



Word	Contents
1	Origin centroid external number p*
2	Number of active destinations (NACT)*
3	Centroid number (internal) of 1st active destination (d1)*
4	Centroid number (internal) of 2nd active destination (d2)*
.	
.	
.	
qa	Centroid number (internal) of qath active destination (dqa)*
.	
.	
.	
NACT + 2	Centroid number (internal) of last active destination (dNACT)
NACT + 3	Value for 1st active destination ($V_{p,d1}$)
NACT + 4	Value for 2nd active destination ($V_{p,d2}$)
NACT + qa + 2	Value for qath active destination ($V_{p,dqa}$)
2 * NACT + 2	Value for last active destination ($V_{p,dNACT}$)

An active destination is a destination (column) which contains a non zero value for the origin p .

* indicates a 2 byte integer in IBM version

FILE NAME : COMFIL File number: 101

CONTENTS : EMME commands, subcommands, header and data

TYPE : Direct entry input file

SYMBOL :

CREATED BY :

INPUT TO : 1.11 RDCOM
databank

FILE FORMAT : Sequential
Formatted
Record length: 80 characters
Each record contains a command, a subcommand, a databank header or dat

RECORD FORMAT : Each record contains 80 characters
Characters 1-72 contain the information for the command, subcommand, header or data as described in the "Command Language Description".
Characters 73-80 are available for user information and are printed when the record is printed.

File name : NODES File number: 200

Identification : file of geocoded nodes (expanded)

Type : data base

Position in index : 5

Created by : 2.110 IGE0

Sorted by : (at creation time)

Record format : each record is 2 words long

1- X-coordinate

2- Y-coordinate

The position of the record in file is the internal node number.

Accessed by : SUBROUTINE GTNODE (INT, IEXT, IX, IY)

Returns the external node number and the coordinates.

The arguments are all of type INTEGER:

INT	Internal node number
IEXT	External node number
IX	X-coordinate
IY	Y-coordinate

File name : LINES File number: 201

Contents : Data describing each transit line

Type : Data base

Input to : 5.12 BLDNET

Index position : 12

Created by : 2.25 ALINE

Updated by : 2.55 MLINE
2.54 DLINE
2.24 BLINE
2.26 ILINE

File format : Sequential
Record L contains description of line
L, L=1, NLINE
Record length = 14 + 2 * KNODE, where KNODE is the number of nodes on the line.

Record format :

Word	Contents
1	Line number
2	Activity
3	Name of the line (5 words)
8	Headway
9	Length
10	Speed
11	Type
12	Lay over time
13	Number of vehicles
14	Number of nodes on the route (KNODE)
14+i	Node number, i odd from 1 to 2 x KNODE
14+j	Direction, j even from 2 to 2 x KNODE

Accessed by : 1) SUBROUTINE PSLINE
PSLINE sets the data base to read from the beginning of file LINES.

2) SUBROUTINE RDLINE (HDW, KNODE, LISTN, NDIR, LEØF)
Reads next line data from file LINES.

The arguments are:

Type	Argument	Will contain
REAL	HDW	Headway (in minutes)
INTEGER	KNODE	Number of nodes in LISTN
INTEGER	LISTN	Vector of nodes[+]
INTEGER	NDIR	Vector of directions[+]
LOGICAL	LEOF	

LEOF = .TRUE. indicates end of file, otherwise LEOF is set to FALS

The line numbers (internal) correspond to the sequence.

2) SUBROUTINE GTLINE(L, NOEXT, NAME , DIST, SPD, LTYP, TLAY, NVH)

Provides a direct access in the data base to the characteristics of line L. The arguments are:

INTEGER	L	Internal line number
INTEGER	NOEXT	External line number
INTEGER	NAME	Name of line (20A1 format)
REAL	DIST	Total length of line
REAL	SPD	Commercial speed of line
INTEGER	LTYP	Line type
REAL	TLAY	Lay over time in minutes
INTEGER	NVH	Number of vehicles

Note: the position of file LINES is not altered by a CALL GTLINE.

NOEXT = 0 if line L does not exist.

Values returned for undefined data are

. blank for NAME

. -1 or -1.0 for the other parameters.

[+] indicates a 2 byte integer in the IBM version.

File name	:	N0DC0R File number: 202
Contents	:	Transit node - road nodes correspondence table
Type	:	Data base
Input to	:	5.14 EXPNET
Index position	:	20
Created by	:	2.14 AMINI
Updated by:	:	2.53 MMINI 2.52 DMINI 2.13 BMINI 2.15 IMINI
File format	:	Sequential 1 record for each entry in correspondence table, sorted in ascending order by transit node Record length = 3 + NB where NB is the number of corresponding road nodes.

Record format :

Word	Contents
1	Transit node number
2	Activity
3	Number of nodes in expansion (NB)
3+i	One of corresponding road nodes, i = 1, NB

Accessed by : 1) SUBROUTINE RDCOR(NODE, NB, LIST, LEOF)

Returns the next entry in the node correspondence table.

The arguments are defined as:

INTEGER NODE	is the transit node number (internal data base node number)
INTEGER NB	is the number of corresponding road nodes
INTEGER LIST	is a vector containing the corresponding road nodes (data base internal node numbers)
LOGICAL LEOF;	LEOF is set to TRUE to indicate end of file, otherwise, it is set to FALSE.

2) SUBROUTINE PSCOR

PSCOR sets the data base to read from the beginning of file N0DC0R.

CA 28/11/78

File name : ACCESS - EGRESS File number: 203

Contents : Transit access and egress links

Type : Data base

Index position : 16 for access links
18 for egress links

Created by : 2.29 ALNK

Updated by : 2.58 MLNK
2.56 DLNK
2.27 BTL

Sorted by : 2.31 ITL

Input to : 4.21 TIMPED
5.50 TASS

File format : Sequential in two sections

1. Access links sorted in ascending order by link origin (centroid)
2. Egress links sorted in ascending order by link destination (centroid)

The file contains one record for each link

Record length = 4 words

Record format : a) Before renumbering

Word	Contents
1	External origin
2	External destination
3	Travel time
4	Activity

b) After renumbering

Word	Contents
1	Internal origin x 2^{14} + internal destination
2	External origin x 2^{14} + external destination
3	Travel time
4	Activity

Accessed by : 1) SUBROUTINE PSACCL

PSACCL sets the data base to read from the beginning of section 1 (access links)

2) SUBROUTINE PSEGRL

PSEGRL sets the data base to read from the beginning of section 2 (egress links)

3) SUBROUTINE RDACCL(ICENT, NODE, TIME, EOF)

RDACCL reads the next access link in section 1

Parameters are defined as:

INTEGER ICENT is the origin node of the link (centroid)

INTEGER NODE is the destination node of the link (transit network node)

REAL TIME is the walking time (minutes) on the link

LOGICAL EOF ; EOF is set to TRUE if there are no more access links (end of section 1), otherwise it is set to FALSE

4) SUBROUTINE RDEGRL(NODE, ICENT, TIME, EOF)

RDEGRL reads the next egress link in section 2

Parameters are defined as above except that NODE is the origin node of the link and ICENT is the destination node of the link. EOF will indicate the end of section 2

NOTES: 1) Both sections are read independently as two separate files

2) Node numbers are internal data base node numbers

File name : TLTIM File number: 204

Contents : User specified travel time for transit-only links

Type : Data base

Index position : 14

Created by : 2.29 ALNK

Updated by : 2.58 MLNK
2.56 DLNK
2.27 BTL

Sorted by : 2.81 ITL

File format : Sequential, sorted in ascending order by link origin and destination node
1 record for each transit only link
Record length = 4 words

Record format : a) Before numbering

Word	Contents
1	External origin
2	External destination
3	Travel time
4	Activity

b) After numbering

Word	Contents
1	Internal origin x 2^{14} + internal destination
2	External origin x 2^{14} + external destination
3	Travel time
4	Activity

Accessed by : 1) SUBROUTINE RDTLT(NORIG, NDEST, TIME, LEOF)

RDTLT reads the next transit link on file TLTIM

Parameters are defined as:

INTEGER NORIG	is the origin node of the link
INTEGER NDEST	is the destination node of the link
REAL TIME	is the transit travel time on the link
LOGICAL LEOF	is set to TRUE to indicate end of file, otherwise it is set to FALSE

Node numbers are data base internal node numbers

a) SUBROUTINE PSTLT

PSTLT sets the data base to read from the beginning of file TLTIM.

File name : INITIM File number: 205

Contents : Dummy file representing either file 211 RLINK or file 631 RLTIVL

Type : Intermediate file

Input to : 5.15 TTIME

File format : Sequential

Record format : Each record contains the origin and destination of a road link, the length of the link and either the observed or predicted time on the link. (See descriptions of files RLINK and RLTIVL and module TTIME).

Accessed by : For each road link in section 2 of RLINK link length and observed time (if available) are read from file RLINK using subroutines PSLINK and RDLINK. If a predicted travel time is to be used (specified by a subcommand parameter) it is read from file RLTIVL using subroutine RDLTV2

File name : RLINK File number: 211

Contents : Data describing each link in the network

Type : Data base

Index position : 6 for normal links
8 for access links
10 for egress links

Created by : 2.18 SRL (User specified)

Input to : 6.41 NETWRK
5.14 EXPNET

File format : Sequential in 3 sections

1. Access links from centroids
2. Egress links to centroids
3. Links between nodes (normal links)

Each section is sorted in ascending order by link origin and destination nodes. The file contains 1 record for each link L=1, NLINK

Record length = 15 words

Record format :

Work	Contents
1	internal origin x 2^{14} + internal destination
2	external origin x 2^{14} + external destination
3	Link type
4	length
5	observed time
6	observed volume
7 to 11	data specified by user (5 words)
12	text (4 words)

Accessed by : 1) SUBROUTINE PSLINK(NS)

PSLINK sets the data base to read from the beginning of the section NS , NS = 1 or 2. NS = 1 indicates Section 1. NS = 2 indicates sections 2 + 3.

2) SUBROUTINE RDLINK(NORIG, NDEST, LTYPE, DIST, TIME, VOLUME,EOF)

RDLINK reads the data for the next link L.

The starting section is determined by a CALL PSLINK(NS).

When the last link of section 1 has been read by a CALL RDLINK, the next CALL RDLINK will read the first record from section 2 + 3. Sections 2 + 3 are sorted together into ascending order by origin and destination nodes.

The arguments of RDLINK are defined as:

Type	Argument	Will contain
INTEGER	NORIG	Link origin node number+
INTEGER	NDEST	Link destination node number+
INTEGER	LTYPE	Link type
REAL	DIST	Link length
REAL	TIME	Observed travel time on link
REAL	VOLUME	Observed volume on link
LOGICAL	EOF	TRUE if the last link on the file has already been read, otherwise it is set to FALSE.

3) SUBROUTINE RLINKU(UDATA, NP, EOF)

RLINKU reads the used defined data for the next link of the current section. The section is determined in the same way as for SUBROUTINE RDLINK. The arguments of RLINKU are defined as:

REAL UDATA(NP). UDATA(N) will contain the value of the N^{th} data for the next link.

+ indicates a 2 byte integer on the IBM version.

INTEGER NP. NP specifies the number of data items the user wishes to read.

LOGICAL EOF. EOF will contain the same values as EOF in SUBROUTINE RDLINK.

The data items are numbered in the same order as they appear in the input data.

Note : The data bank buffer for the file is released when EOF is set to TRUE, i.e. the last link on the file has already been read when the CALL RDLINK of CALL RLINKU is executed.

File name : PERSPQ File number: 221

Contents : OD matrix of trips (persons)

Type : Data base

Index position : 28 for matrix # 1
29 for matrix # 2
30 for matrix # 3

Symbol : $\bar{g}_{pq}$

Created by : 2.37 AODMAT

Updated by : 2.100 MODMAT
2.102 DODMAT
2.34 BMAT

Sorted by : 2.39 IMAT

Input to : 6.10 AUTRIP
5.41 ODTRAN
5.50 TASS

File format : Sequential

1 origin record for each active origin centroid plus 1 column record for each non zero entry in the matrix.

Record length = 3 words

Record format : 1. Active origin records

Word	Contents
1	Origin node number p (internal)
2	Record number (block number) of first column record for the origin p
3	Active indicator

2. Column records

Word	Contents
1	Destination node number q
2	Value of entry in matrix $\bar{g}_{pq}$
3	Active indicator

Accessed by : 1) SUBROUTINE PSOD

PSOD sets the data base to read from the beginning of the current O-D MATRIX file, i.e. to read the line for origin 1.

2) SUBROUTINE RDOD (TRIP, NORIG, NBDEST, EOF)

RDOD reads the next line of the O-D MATRIX, i.e. trips from origin p . The arguments are defined as:
REAL TRIP (NCENT). TRIP(ND) will contain the number of trips from origin p to destination number ND. (p and ND are the internal numbers of the next origin and each destination).

INTEGER NORIG. NORIG will contain the origin number p .

INTEGER NBDEST. NDBEST will contain the number of non zero values in the vector TRIP.

LOGICAL EOF. EOF is set to TRUE if the last line in the O-D MATRIX has already been read, otherwise it is set to FALSE. The O-D MATRIX contains NCENT lines.

Note : The data bank buffer for the file is released when EOF is set to TRUE.

File name : AUTOCC File number: 222

Contents : Auto occupancy between each origin and destination centroid

Type : Data base

Index position : 31

Symbol : γ_{pq}

Created by : 2.37 AODMAT (User specified)

Updated by : 2.100 MODMAT
2.102 DODMAT
2.34 BMAT

Input to : 6.10 AUTRIP
5.41 ODTRAN

File format : See file number 221.

Record format : See file number 221.

Accessed by : 1) SUBROUTINE PSOCC

PSOCC sets the data base to read from the beginning of the auto occupancy matrix.

2) SUBROUTINE RDOCC (OCCUP, NORIG, NDBEST, EOF)

RDOCC reads the next line (origin p) of the auto occupancy matrix. The arguments are defined as:
REAL OCCUP(NCENT). OCCUP(ND) will contain the auto occupancy for trips from origin p to destination ND (p and ND are the internal numbers of the origin and each destination).

INTEGER NORIG. NORIG will contain the origin number p .

INTEGER NDBEST. NDBEST will contain the number of non zero values in the vector OCCUP.

LOGICAL EOF. EOF is set to TRUE if the last line in the auto occupancy matrix has already been read, otherwise it is set to FALSE. The auto occupancy matrix contains NCENT lines.

Note : The data bank buffer for the file is released when EOF is set to TRUE.

File name : MODEST File number: 223

Contents : Modal split estimate of trips for demand between each origin and destination. The estimate specified as a percentage of the total trips (μ_{pq}) or as an estimate of the number of automobile trips (g^{au}_{pq}).

Type : Data base (Index position = 32)

Symbol : μ_{pq} ii subcommand parameter SPLT = PCNT
g^{au}_{pq} if subcommand parameter SPLT = DEMA

Created by : 2.37 AODMAT (User specified)

Updated by : 2.100 MODMAT
2.102 DODMAT
2.34 BMAT

Sorted by : 2.39 IMAT

Input to : 6.15 INITDM
5.41 ODTRAN

File format : See file number 221.

Record format : See file number 221.

Accessed by : 1) SUBROUTINE PSMEST

PSMEST sets the data base to read from the beginning of the modal split estimate matrix.

2) SUBROUTINE RDMEST (MEST, NORIG, NBDEST, EOF)

RDMEST reads the next line (origin p) of the modal split estimate matrix. The arguments are defined as REAL MEST(NCENT). MEST(ND) will contain the modal split estimate for trips from origin p to destination ND (p and ND are internal numbers of the origin and each destination).

INTEGER NORIG. NORIG will contain the origin number p .

INTEGER NBDEST. NBDEST will contain the number of non zero values in the vector MEST.

LOGICAL EOF. EOF is set to TRUE if the last line in the modal split estimate matrix has already been read, otherwise it is set to FALSE. The modal split estimate matrix contains NCENT lines

Note : The data bank buffer for the file is released when EOF is set to TRUE.

File name : VEHADJ File number: 224

Contents : Vehicle adjustment matrix

Type : Data bank (Index position = 60)

Symbol : e^{car}_{pq}

Created by : 2.37 AODMAT

Updated by : 2.100 MODMAT
2.102 DODMAT
2.34 BMAT

Sorted by : 2.39 IMAT

Input to : 6.10 AUTRIP
6.15 INITDM
6.60 RASS

File format : See file number 221

Record format : See file number 221

Accessed by : 1) SUBROUTINE PSODEX

PSODEX sets the data base to read from the beginning of the vehicle adjustment matrix.

2) SUBROUTINE RDODEX (EXODIJ, NORIG, NBEX, EOF)

RDODEX reads the next line of the vehicle adjustment matrix i.e. vehicles from origin p . The arguments are defined as: REAL EXODIJ(NCENT). EXODIJ(ND) will contain the number of vehicles travelling from origin p to destination number ND (p and ND are the internal numbers of the next origin and each destination).

INTEGER NORIG. NORIG will contain the origin number p .

INTEGER NBEX. NBEX will contain the number of non zero values in the vector EXODIJ.

LOGICAL EOF. EOF is set to TRUE if the last line in the vehicle adjustment matrix has already been read, otherwise it is set to FALSE. The vehicle adjustment matrix contains NCENT lines.

Note: The data bank buffer for the file is released when EOF is set to TRUE.

File name : DCOK File number: 231

Contents : Demand function travel time coefficients

Type : Data base (Index position = 56)

Symbol : $k0_p$, $k2_p$, $k3_p$

Created by : 2.86 IDFTTC (User specified)

Used by : 5.30 CALKTR
5.50 TASS
6.60 RASS

File format : Sequential

Record p contains values $k0_p$, $k2_p$, $k3_p$

The file contains 1 record for each origin p ; p = 1, NCENT.

Record length = 3 words

Record format :

Word	Contents
1	$k0_p$
2	$k1_p$
3	$k2_p$

Accessed by : 1) SUBROUTINE PSDCOK

PSDCOK sets the data base to read from the beginning of the demand function coefficients file

2) SUBROUTINE RDDCOK (K,NORIG,EOF)

RDDCOK reads the coefficients for the next origin p . The arguments are defined as
REAL K (4). K(1), K(2), K(3) and K(4) contain $k0_p$, $k1_p$, $k2_p$, $k3_p$ respectively. (If more than 4 coefficients are required the size of K will be increased).

INTEGER NORIG. NORIG is set to the value of p .

LOGICAL EOF. EOF is set to TRUE if the last set of coefficients on the demand function coefficient file has already been read. Otherwise it is set to FALSE. The demand function coefficient file contains NCENT sets.

Note : The data bank buffer for the file is released when EOF is set to TRUE.

File name: DCOC File number: 233 (part of 359)

Contents : Demand function coefficients applicable to socio-economic variables

Type : Data base (index position = 59)

Symbol : c_i or c_{ip} , c_{iq} , c_{ipq}

Created by : See file description 359 SOCIO

Input to : 7.10 DC0S01

File format: Sequential
1 record for each type A coefficient containing the value for each zone p or q , followed by NCENT records for each type B coefficient. Within each block of NCENT records, record p contains the values of c_{ipq} to all destinations q , q = 1 , NCENT

Record Format: For Type A coefficients, i.e. those coefficients which vary either by origin (c_{ip}) or destination zone (c_{iq}) but not both. Each record contains

- Coefficient number (i)
- Origin/destination indicator
  = b or 0 if coefficient is for origin zone
  = -1 if coefficient is for destination zone
- Value of coefficient for zone 1

  ⋮
- Value of coefficient for zone n (c_{in})
- Value of coefficient for NCENT ($c_{i,NCENT}$)

For type B coefficients, i.e. which vary by origin and destination zones (c_{ipq}). Record p in each group contains:

- Coefficient number (i)
- Origin number (p)
- Value of coefficient for destination 1 (c_{ip1})
- Value of coefficient for destination n (c_{ipn})

  ⋮
- Value of coefficient for destination ($c_{ip,NCENT}$)

Note: Files DCOC and DVAR must be sorted into the same sequence

Accessed by : SUBROUTINE PSDC0C

PSDCOC sets the data base to read from the beginning of the demand function coefficient file DCOC.

SUBROUTINE RDDC0C (ICOEF,NORIG,COEF,EOF)

RDDOC reads the next set of demand function coefficients.

The arguments are defined as follows

INTEGER ICOEF. ICOEF will contain the number i of the demand function coefficient c_i read.

INTEGER NORIG will contain either

a) origin/destination indicator if the coefficient is type A.

b) origin zone number of the set of values read, if the coefficient is type B.

REAL COEF (NCENT). COEF is filled by subroutine RDDC0C according to the type of coefficient (A or B).

For type A coefficients, COEF (IP) will contain the value of coefficient c_{IVAR} for zone IP.

For type B coefficients, COEF (IQ) will contain the value of c_{IVAR} between origin zone NORIG and destination zone IQ.

LOGICAL EOF: EOF is set to TRUE if the last record on the demand function coefficient file has already been read, otherwise it is set to FALSE.

Note : The data bank buffer for the file is released when EOF is set to TRUE.

File name: DVAR File number: 234 (Part of 359)

Contents : Values of socio-economic variables in the demand function

Type : Data base (Index position = 59)

Symbols : x_i or x_{ip} , x_{iq} , x_{ipq}

Created by : See file description 359 SOCIO

Input to : 7.10 DC0S01

File Format : Sequential
Same format as file DCOC

Record Format: Same formats as for file DCOC

Note : Files DCOC and DVAR must be accessed in the same sequence

Accessed by : SUBROUTINE RDDVAR(IVAR,NORIG,VAR,EOF)

RDDVAR reads a set of socio-economic variables.

The arguments are defined as follows

INTEGER IVAR. IVAR contains the number i of the socio-economic varible x_i to be read.

INTEGER NORIG. NORIG contains either

a) zero if the variable is Type A corresponding with a Type A coefficient (see description of file 233 DCOC)

b) origin zone number of the set of variables to be read, for Type B variables which correspond with Type B coefficients.

REAL VAR (NCENT). VAR is filled by subroutine RDDVAR according to the type of variable (A or B). For type A variables, VAR(IP) will contain the value of the variable x_{IVAR} for zone IP.

For type B variables, VAR(IQ) will contain the value for x_{IVAR} between origin NORIG and destination IQ.

LOGICAL EOF. EOF is set to TRUE if there is no data for the required variable.

Note that if IVAR specifies a variable which is type B and NORIG is zero, RDDVAR will read the values for the first origin into VAR and will set NORIG to that origin number.

File name	:	PMUW File number: 250
Contents	:	Transit path coefficients multipliers for origin p $w1_p$ multiplies walk time coefficient $w2_p$ multiplies wait time coefficient
Type	:	Data base (Index position = 63)
Symbol	:	$w1_p$, $w2_p$
Created by	:	2.117 ITPM (User specified)
Used by	:	4.21 TIMPED 5.50 TASS
File format	:	Sequential Record p contains values $w1_p$, $w2_p$ for origin p (2 words). The file contains 1 record for each origin p; p = 1, NCENT

Record format :

Word	Contents
1	Value of $w1_p$
2	Value of $w2_p$

Accessed by :

1) Subroutine PSPMUW

PSPMUW sets the data base to read from the beginning of the multipliers file.

2) Subroutine RDPMUW (W, NORIG, LEOF)

RDPMUW reads the multipliers for the next origin p.

The arguments are defined as

REAL W(2). W(1) contains $w1_p$ and W(2) contains $w2_p$.

INTEGER NORIG . NORIG is set to the value of p.

LOGICAL LEOF. LEOF is set to TRUE is there are no more sets of multipliers on the file, otherwise it is set to FALSE.

NOTES : 1) The data base checks that

$w1_p \geq 0.0$

$w2_p \geq 0.0$

2) If, for one p, the multipliers have not been defined (blank), the subroutine RDPMUW will return ones (1.0) in W.

File name: CENT File number: 302

identification : file of centroids

type : data base

position in index : 2

created by : 2.12 ACEN

updated by : 2.51 MCEN 2.50 DCEN 2.10 BCEN

sorted by : 2.11 TCEN

record format : each record is 8 words long

1- number of centroid

2- active indicator

3- X - coordinate

4- Y - coordinate

5- name of zone (4 words)

File name : NODEIN File number: 304

Identification : File of geocoded nodes (raw data)

Type : Data base

Position in index : 4

Created by : 2.106 AGEO

Updated by : 2.107 DGEO

2.108 MGEO

2.109 BGEO

Sorted by : Not sorted

Record format : Each record is 4 words long

1- node number

2- active indicator

3- X-coordinate

4- Y-coordinate

File name : DBSTAT File number: 336

Identification : file of status

Type : data base

Position in index : 36

Created by : 3.21 OPBAN

Updated by : 2.119 IJ

2.62 B

Record format : the first 20 words are vector beta (status of data bank) and the next 20 words are vector gamma (indicates if a scenario has changed a file in data bank)

Values for beta are:

+1 if an error was detected in the last validation

0 If no error was detected

-1 if a correction is performed in the current job

Values for gamma are:

+1 if a scenario was applied in a previous job

0 if no scenario is applied

-1 if a scenario is being applied in this job

These values are set for each entry, i.e.

1- centroids
2- road links
3- transit only nodes
4- transit lines
5- transit links
6- correspondence table
7- O-D matrix #1
8- O-D matrix #2
9- O-D matrix #3

10- vehicle adjustment matrix
11- auto occupancy matrix
12- modal split matrix
13- socio-economic data
14- demand function coefficients
15- transit path multipliers
16- geocoded nodes
17- zone grouping
18- scale
19-
20-

File name: STOPS File number: 337

identification : file of nodes encountered on transit lines

type : data base

position in index : 37

created by : 2.25 ALINE

updated by : 2.55 MLINE, 2.54 DLINE, 2.22 BTRO

sorted by : not sorted

usage : to create the list of transit-only nodes in 2.23 ITRO

record format : each record is 3 words long

1- external node number
2- frequency on permanent transit lines
3- frequency on temporary transit lines

File name: RLNKIN File number: 339

identification : file of road links

type : data base

position in index : 39

created by : 2.21 ARL

updated by : 2.49 MRL, 2.48 DRL, 2.17 BRL

sorted by ; not sorted

sequence : random

usage : to create a correct RLINK in a reasonable amount of time.

record format : each link is 16 words long

1- origin
2- destination
3- active indicator
4- type
5- length
6- observed time
7- observed volume
8 to 12- data specified by user (5 words)
13- text (4 words)

File name: NODEXT File number: 354

identification : file of internal-external numbers

type : data base

position in index : 54

created by : 2.60 KEEPND

updated by :

sorted by : not sorted

record format : file contains 4 records.
each one has the same structure

1- number of nodes of a given type (N)
2- external node number
. "
. "
N+1 "

note : the position of a node in a vector containing the 4 records is the internal number.

Accessed by: 1) SUBROUTINE GTNODE (INT,EXT,X,Y)

GTNODE gets the external node number which corresponds to a given internal node number and its coordinates if they are specified.

The arguments are defined as follows:

INTEGER INT	Internal node number (Input)
INTEGER EXT	External node number (Output) If INT does not exist EXT is set to zero
REAL X	X coordinate of INT
REAL Y	Y coordinate of INT If INT does not exist X and Y are set to infinity.

2) SUBROUTINE GTNODV (ISTART, IEND, EXTVEC)

GTNODV gets a vector of external node numbers corresponding to internal node numbers ISTART through IEND.

The arguments are defined as follows:

INTEGER ISTART	Internal node number of first node in the sequence (Input)
INTEGER IEND	Internal node number of last node in the sequence (Input)
INTEGER EXTVEC	Vector to be filled with the external node numbers EXTVEC(1) will contain the external node number for internal node number ISTART

IEND must be greater than or equal to ISTART. Otherwise EXTVEC will not be filled. EXTVEC must be a vector dimensioned at least IEND-ISTART+1.

3) SUBROUTINE PTNODE (EXT,INT)

PTNODE gets the internal node number corresponding to a given external node number.

The artuments are defined as follows:

INTEGER EXT	External node number (Input)
INTEGER INT	Internal node number (Output) If EXT does not exist, INT is set to zero.

File name: DCOKIN .. File number: 355

identification : file of demand function travel time coefficients (raw data)

type : data base

position in index : 55

created by : 2.65 ADFTTC

updated by : 2.67 MDFTTC, 2.66 DDFTTC, 2.68 BDFTTC

sorted by : not sorted

sequence : sequential

usage : to create file DCOK in module 2.86 IDFTTC

record format : each record is 5 words long

1 - zone number of name of group or "all"
2 - active indicator
3 - KO constant
4 - K1 auto time
5 - K2 transit time

File name : ZGROUP File number: 357

identification ; file of zone grouping

type : data base

position in index : 57

created by : 2.69 AZONEG

updated by : 2.70 DZONEG, 2.71 BZONEG

sorted by : not sorted

sequence : sequential with index

record format : a) section of name of group

1- name of the group
2- active indicator
3- beginning of the list of nodes in the group (block number x (length of a block + 1) + position within the block)

b) section of definition of groups: each group is defined by some chained partial lists

1- number of elements in this partial list (N)
2- zone number } repeated N times
3- activity }
.
.
.
2N-
2N+1-
2N+2- pointer to the next sequence or 0; format is the same as in (3-) of first section.

File name: SOCIO File number: 359

identification : file of socio-economic data

Note: This file contains the information for files DCOC and DVAR

type : data base

position in index : 59

created by : 2.73 ASE, 2.81 ASE2, 2.76 AKEY

updated by: 2.74MSE 2.75 DSE, 2.77 DKEY, 2.80 MSE2, 2.79 DSE2, 2.84 BSE

sorted by : 2.82 ISE

sequence : sequential with index

record format:

a) first section (index)

. zone keys variables

$$\left.\begin{array}{l}\text{- name}\\ \text{- origin or destination, 1 char.}\\ \text{- \# of block and position within block (expanded data)}\\ \text{- \# of block (raw data)}\\ \text{- permanent value for "all"}\\ \text{- temporary value for "all"}\end{array}\right\}\begin{array}{l}\text{repeated 8 times,}\\ \text{as there are 8 keys}\end{array}$$

. zone keys coefficients

$$\text{same as above}\;\Big\}\;\text{repeated 8 times}$$

. interzone keys variables

- name
- (void)
- # of blocks and position within block (expanded data)
- # of block (raw data)
- permanent value for "all-all"
- temporary value for "all-all"

$\Big\}$ repeated 8 times

. interzone keys coefficients

same as above } repeated 8 times

b) second section . Raw data

. "zone" variables and coefficients

- origin (either a zone or a group)
- active indicator
- value

. "interzone" variables and coefficients

- origin
- destination
- active indicator
- value

c) third section . Expanded data

. "zone" variables and coefficients

If the number of non-zero entries is greater than NCENT/2, we use the following format

1- NCENT
2- value_1
3- value_2
.
.
.
NCENT+1- $\text{value}_{\text{NCENT}}$

If the number of non-zero entries is less than NCENT/2, we use a compressed format

- -NB (the number of non-zero entries with minus sign)
- position }
- value } NB times

. "interzone" variables and coefficients }
  - origin }
  - expanded or compressed vector as in the "zone" case } repeated for each origin
  - 0 to indicate the end of the matrix }

Note that these vectors or matrices are repeated for each key of variables and coefficients.

Accessed by: Subroutines PSDCOC }
RDDCOC } see file 233 DCOC

PSDVAR }
RDDVAR } see file 234 DVAR

File name : SCEN File number: 361

identification ; file of scenarios

type: data base

position in index : 61

created by : 2.98 SCENO

updated by : 2.99 ACCUM

sorted by : not sorted

sequence : sequential with index

record format :

a) first section . Index
   1- name of scenario (2 words)
   2- number of block of the accumulated card images.

b) second section . Card images
   1- number of characters in the next card images
   2- packed card image, as many characters per word as possible and trailing blanks removed. (Needs a variable number of words).

File name : PMUWIN File number: 362

Identification : File of transit path multipliers (raw data)

Type : Data base

Position in index : 62

Created by : 2.113 ATPM

Updated by : 2.114 DTPM

2.115 MTPM

2.116 BTPM

Sorted by : Not sorted

Sequence : Sequential

Record format : Each record is 4 words long

1- zone number or name of group of "all"

2- active indicator

3- W1 transit access time

4- W2 transit waiting time

File name : TLLIST File number: 501

Contents : Vectors representing the transit network structure

Type : Stored in common area /STRUC/

Created by : 5.12 BLDNET

Input to : 5.14 EXPNET

5.15 TTIME

File format : Data stored in common /STRUC/ are:

FLIN(L), L = 1, NLINE

DA(J) , J = 1, NSEG

DR(J) , J = 1, NSEG

NOLIST*(L), L = 1, NLINE

NONLIN*(L), L = 1, NLINE

LISTN*(J) , J = 1, NSEG

NDIR*(J) , J = 1, NSEG

LISTL*(K) , K = 1, NSEL

LPNT*(K) , K = 1, NSEL

NADR*(K) , K = 1, NSEL

See also file TNET (515) and modules 5.12 BLDNET, 5.15 TTIME

Note: Node numbers are transit internal node numbers.

---

* two byte words in IBM version

File name : DC0KTR File number: 511

Contents : Component of demand function which is independent of transit impedances

Type : Work file

Symbol : K_{pq}^{tr}

Created by : 5.30 CALKTR

Input to : 5.50 TASS

File format : Sequential

Binary

Record length = NCENT words

Record p contains the values K_{pq}^{tr} , $q = 1$, NCENT

File contains 1 record for each origin p , $p = 1$, NCENT

Record format :

Word	Contents
1	Value of K_{p1}^{tr} ; origin p , destination 1
2	Value of K_{p2}^{tr} , origin p , destination 2
.	
.	
.	
q	Value of K_{pq}^{tr} ; origin p , destination q
.	
.	
.	
NCENT	Value of $K_{p,NCENT}^{tr}$; origin p , destination NCENT

Use : Read in 5.50 TASS to calculate variable transit demand

File name	:	CØRNET		File number: 514	
Contents	:	Vectors representing the transit line itineraries as sequences of road links, including turning links			
Type	:	Intermediate file			
Created by	:	5.14 EXPNET			
Input to	:	4.18 BUSFRQ 4.31 TIMCHK			
File format	:	Sequential Binary 6 records for each line L , L = 1, NLINE			

Record format :

	Record	Words+	Contents
	1	3	NT, NA, NR
IF NT > 0	2	NT	NOT(I), I = 1, NT
IF NA > 1	3	NA	NOA(I), I = 1, NA
	4	NA-1	LINKA(I), I = 2, NA
IF NR > 1	5	NR	NOR(I), I = 1, NR
	6	NR-1	LINKR(I), I = 2, NR

See module 5.14 EXPNET for description of vectors

Use : To establish correspondence between road and transit networks

Note : On file CORNET node numbers are data-base internal node numbers.

+ two byte words in IBM version

File name : TNET File number: 515

Contents : Vectors representing the transit network structure

Type : Intermediate file

Created by : 5.15 TTIME

Input to : 4.18 BUSFRQ
4.21 TIMPED
4.31 TIMCHK (also updated)
4.61 CAPCHK (also updated)
5.50 TASS

File format : Sequential
Binary
11 records

Record format :

Record	Words	Contents
1	3	NLINE, NSEG, NSEL
2	NLINE	FLIN(L) , L = 1, NLINE
3	NLINE[+]	NOLIST(L) , L = 1, NLINE
4	NLINE[+]	NONLIN(L) , L = 1, NLINE
5	NSEG[+]	LISTN(J) , J = 1, NSEG
6	NSEG[+]	NDIR(J) , J = 1, NSEG
7	NSEG	DA(J) , J = 1, NSEG
8	NSEG	DR(J) , J = 1, NSEG
9	NSEG	DISA(J) , J = 1, NSEG
10	NSEG	DISR(J) , J = 1, NSEG
11	NSEL[+]	LISTL(K) , K = 1, NSEL
12	NSEL[+]	LPNT(K) , K = 1, NSEL
13	NSEL[+]	NADR(K) , K = 1, NSEL

See module 5.15 TTIME for description of DA and DR

[+] two byte words in IBM version

Use : 1) Vector FLIN is used to calculate bus frequency by road link in 4.18 BUSFRQ

2) To calculate minimum impedance paths in modules 4.21 TIMPED and 5.50 TASS

3) To compare and update transit travel times in 4.31 TIMCHK

4) To compare and update headways in 4.61 CAPCHK

File name : TLFREQ File number: 518

Contents : Number of transit vehicles per hour on each road link used by transit

Type : Intermediate file

Symbol : f_a^{tr}

Created by : 4.18 BUSFRQ

Input to : 6.50 BUSIN

File format : Sequential

Binary

Record length = variable

Record format :

Word	Contents
1	NORIG : road link origin node
2	NDEST : road link destination node
3	FATR : number of buses on link
4	NL : number of transit lines on link
5 . . . NL+4	LILIST: list of transit lines on link

FATR is of type REAL.

Use : To define transit vehicle volume on road links.

Note : The file may be empty of there are no transit vehicles on any road links.

File nale : TTREES File number: 521

Contents : Shortest path tree from each origin p to all transit network nodes n , n=1,NNODET

Type : Work file

Created by : 4.21 TIMPED

Input to : 5.50 TASS

File format : Sequential

Binary

Record length = 3*NNODET

Number of records = one for each origin p which is linked to the transit network (i.e. for which the tree actually exists)

Record format :

Word	Contents
1	TIAC(1)
⋮	
NNODET	TIAC(NNODET)
NNODET+1	NOAC(1)
⋮	
2*NNODET	NOAC(NNODET)
2*NNODET+1	LIAC(1)
⋮	
3*NNODET	LIAC(NNODET)

TIAC is of type REAL, NOAC, LIAC are of type INTEGER

TIAC(N) = Time (minutes) from origin to node N

NOAC(N) = Predecessor node of node N on the path from origin to node N

LIAC(N) = Transit line used from node NOAC(N) to node N

Use : Module 5.50 TASS, in the case of bimodal assignment only, will read the transit paths calculated by module 4.21 TIMPED in the last iteration of the bimodal procedure. File TTREES is located on the same logical unit as file DCOKTR.

CA 14/09/78

File name : TZIMP File number: 531

Contents : Transit impedance

Type : Intermediate file

Symbol : u^{tr}_{pq}

Created by : 4.31 TIMPED

Input to : 6.25 CALCK

File format : Random

Binary

Record length = NCENT words

Record p contains values u^{tr}_{pq} , $q = 1$, NCENT

File contains 1 record for each origin p , $p = 1$, NCENT.

Record format :

Word	Contents
1	Value of u^{tr}_{pq} (origin p, destination 1)
2	Value of u^{tr}_{pq} (origin p, destination 2)
q	Value of u^{tr}_{pq} (origin p, destination q)
NCENT	Value of $u^{tr}_{p,NCENT}$ (origin p, destination NCENT)

File name : TZTIM File number: 532

Contents : Four components of predicted interzone transit travel times

Type : Intermediate file

Symbol : T^{tr}_{pq}

Created by : 5.50 TASS

Input to : Reports

File format : Sequential

Binary

Record length = NCENT + 1 words (except record 1)

Number of records = NCENT + 1

Record p + 1 contains NODEP, (T^{tr}_{pq} q = 1, NCENT)

where

$$T^{tr}_{pq} = NSEC*10^7 + T^{tr}_{pq}(acc)*10^5 + T^{tr}_{pq}(w)*10^3 + T^{tr}_{pq}(v)$$

NSEC = number of line sections from p to q

$T^{tr}_{pq}(acc)$ = access + egress time (minutes)*

$T^{tr}_{pq}(w)$ = waiting time (minutes)*

$T^{tr}_{pq}(v)$ = in-vehicle time (minutes)*

Record 1 is an identification record

Record formats : a) Identification record

Word	Contents
1	Run id day
2	time
3	Dummy
4-13	Text (40 chars., 4 chars. per word)

b) Origin records, NCENT + 1 words.

Word q + 1 contains T^{tr}_{pq} for destination q , q = 1, NCENT

Word 1 contains NODET, the external node number of centroid p.

---

* Rounded integer.

Use : For reports, $T^{tr}_{pq} = T^{tr}_{pq}(acc) + T^{tr}_{pq}(w) + T^{tr}_{pq}(v)$

Impedance:

$$u^{tr}_{pq} = w1_p*WALK*T^{tr}_{pq}(acc) + w2_p*WAIT*T^{tr}_{pq}(w) + T^{tr}_{pq}(v) + WPEN*NSEC$$

Notes :- NODEP and T^{tr}_{pq} are of type INTEGER.

- T^{tr}_{pp} is set to 0 (zero), p=1, NCENT
- T^{tr}_{pq} is set to 9999999999 if there is no path from p to q.

File name : GPQTR File number: 541

Contents : Matrix of transit trips g_{pq}^{tr}

Type : Intermediate file

Symbol : g_{pq}^{tr}

Created by : 1) 5.41 ODTRAN (transit assignment with fixed demand)

2) 5.50 TASS (transit assignment with variable demand)

Input to : 1) 5.50 TASS (fixed demand only)

2) Reports

File format : Sequential

Binary

Record length (variable) = 2 * $NACT_p$ + 2 ; $NACT_p$ = number active destinations, origin p .

For IBM , length = 6 * . $NACT_p$ + 4 bytes.

The file contains 1 record of identification followed by 1 record for each origin p , p = 1, NCENT

Record formats : a) Identification record

Word	Contents
1	Run id day
2	time
3	Dummy
4-13	Text (40 chars., 4 chars. per word)

b) Origin records, one record for each origin p, p = 1, NCENT in following format

Origin	# destns	Numbers of active destinations	Predicted demand to active destinations

See also "Packed OD matrix file format"

CA 24/10/77

File name : TFLOW File number: 550

Contents : Transit assignment statistics and predicted passenger volumes on transit network

Type : Intermediate file

Created by : 5.50 TASS

Input to : 1) 4.61 CAPCHK

2) Reports

File format : Sequential in two sections

Binary

Record length = variable (see record formats)

Number of records:

. identification = 1

. section 1 = NCENT + 1

. section 2 = 2 * NLINES

Record format : 1) Identification record

Word	Contents
1	Run id day
2	time
3	Dummy
4-13	text (40 chars., 4 chars. per word)

2) Section 1 (origin statistics)

Record 1 contains NCENT (1 word)

Record p+1 contains (9 words) (p=1,NCENT)

NODEX,TOTOR,FLOT,TMS,TINV,TATM,TACM,DINV,ANST

NODEX is of type INTEGER, all others are of type REAL

NODEX = External centroid number

TOTOR = Total demand originating at centroid p in O-D matrix GPQTR

FLOT = Actual number of trips originating at p that have been assigned on the transit network (trips for which there is no path and intra-zone trips are not assigned)

TMS = Mean total trip time for passengers originating at p

TINV = Mean in-vehicle time for passengers originating at p

TATM = Mean waiting time for passengers originating at p

TACM = Mean access-egress time for passengers originating at p

DINV = Mean trip distance (miles) on the network for passengers originating at p (access-egress not included)

ANST = Mean number of line sections used by passengers originating at p (= mean number of transfers + 1)

3) Section 2 (Passenger volumes)

Record 2*L-1 contains DY(J) , J = 1, KSEG

Record 2*L contains DZ(J) , J = 1, KSEG

where

DY is the array of volumes on forward itinerary of line L

DZ is the array of volumes on reverse itinerary of line L

KSEG = NONLIN(L) - NOLIST(L) + 1

(see file TNET)

DY and DZ are similar to DA and DR on file net

Use : Before using TFLØW for reports, the corresponding TNET file must be created (see 5.15 TTIME).

File name : GBPQ File number: 601

Contents : Vehicle equivalent of total person trip matrix

Type : Work file

Symbol : $\bar{g}^{veh}_{pq}$

Created by : 6.10 AUTRIP

Input to : 6.21 INITDM
6.60 RASS

File format : Sequential
Binary
Record length = NCENT words
Record p contains values of $\bar{g}^{veh}_{pq}$ for origin p
File contains 1 record for each origin p, p = 1, NCENT.

Record format :

Word	Contents
1	Value of $\bar{g}^{veh}_{pq}$ (origin p, destination 1)
2	Value of $\bar{g}^{veh}_{pq}$ (origin p, destination 2)
⋮	
q	Value of $\bar{g}^{veh}_{pq}$ (origin p, destination q)
⋮	
NCENT	Value of $\bar{g}^{veh}_{p,NCENT}$ (origin p, destination NCENT)

Use :
1. In calculation of OD matrix of automobile trips , given modal split estimate on file MODEST, or a default value. (module 6.21 INITDM)
2. In calculation of implied impedance (inverse of demand function (module 6.60 RASS).

File name : GPQRAN File number: 602

Contents : Demand matrix for mode-automobile (vehicles)

Type : Work file

Symbol : g_{pq}^{car}

Created by : 6.15 INITDM (initialized)

Input to : 6.60 RASS (also updated)

File Format : Random (CDC Mass Storage; IBM Direct Access)

Binary

Record length = NCENT words

Record p contains the current values of g_{pq}^{car} for origin p

Records are accessed by numeric index, the origin p

File contains 1 record for each origin p, p = 1, NCENT.

Record format :

Word	Contents
1	Current value of g_{p1}^{car}; origin p, destination 1
2	Current value of g_{p2}^{car}; origin p, destination 2
.	
q	Current value of g_{pq}^{car}; origin p, destination q
.	
.	
NCENT	Current value of $g_{p,NCENT}^{car}$; origin p,destination NCE

Use : The records in file GPQRAN are updated once for each iteration of variable demand assignment in module 6.60 RASS.

File name : DC0KAU File number: 611

Contents : Component of demand function which is independent of automobile impedances

Type : Work file

Symbol : K^{au}_{pq}

Created by : 6.25 CALCK

Input to : 6.60 RASS

File format : Sequential

Binary

Record length = NCENT words

Record p contains the values K^{au}_{pq} , q = 1, NCENT

File contains 1 record for each origin p, p = 1, NCENT.

Record format :

Word	Contents
1	Value of K^{au}_{p1} ; origin p, destination 1
2	Value of K^{au}_{p2} ; origin p, destination 2
.	
.	
.	
q	Value of K^{au}_{pq} ; origin p, destination q
.	
.	
.	
NCENT	Value of $K^{au}_{p,NCENT}$; origin p, destination NCENT

Use : The complete file DC0KAU is read once to obtain the data necessary to a) calculate the implied impedances u^{au}_{pq} (function DEMCST) for q = 1, NCENT; b) accumulate the variable component of the objective function (function TDCST) for q = 1, NCENT. The file is thus rewound and reread once during the initialization and twice during each iteration of the variable demand traffic assignment procedure. (module 6.60 RASS)

File name : DIRFIL File number: 621

Contents : Direction vectors for current iteration of variable demand traffic assignment

Type : Work file

Symbol : d_{pq}

Created by : 6.60 RASS

Input to : 6.60 RASS

File format : Sequential
Binary
Record length = NCENT words
Record p contains the values of d_{pq} ; q = 1, NCENT
File contains 1 record for each origin p ; p = 1, NCENT.

Record format : Word q in each record contains the value of d_{pq}

Use : The complete file is written when the adjustment to the demand is calculated. It is read to obtain this adjustment when the variable demand component of the objective function is calculated and again to adjust the demand. The file is therefore written once and then rewound and read twice during each iteration of the variable demand traffic assignment procedure. (module 6.60 RASS)

File name : RLTIVL File number: 631

Contents : Predicted link volumes and travel times

Type : Intermediate file

Symbol :

Created by : 6.60 RASS

Input to : 4.31 TIMCHK

reports

File format : Sequential

Binary

Record length = 4 words (12 bytes IBM) except record 1

The file contains 1 identification record followed by 1 record for each link in the same sequence as the file RLINK used as input.

Record formats : a) Identification record

Word	Contents
1	Run id e.g. day
2	time
3	Number of iterations
4-13	Text (40 chars., 4 per word)

b) Link records (Record 2 to NLINK + 1)

Word	Contents
1	Origin node number of link (2 bytes IBM)
2	Destination node number of link (2 bytes IBM)
3	Predicted volume for link
4	Predicted travel time for link
5	Auto equivalent of bus volume or link.

Use : 1. To produce reports

2. In restarting the assignment procedure (module 6.60 RASSO

3. To calculate transit travel time on link (module 4.31 TIMCHK)

LGJL 19/12/78

File name : RZTIM   File number: 632

Contents : Predicted interzone travel times

Type : Intermediate file

Symbol : T_{pq}^{au}

Created by : 6.60 RASS

Input to : 5.30 CALKTR

reports

File format : Sequential

Binary

Record length = NCENT words.

The file contains 1 record of identification followed by 1 record for each origin p ; p = 1, NCENT.

Record formats : a) Identification record

Word	Contents
1	Run id day
2	time
3	Number of iterations
4-13	Text (40 chars., 4 chars. per word)

b) Origin records; one record for each origin p, p = 1, NCENT in following format

Word	Contents
1	External node number of origin p
2	Value of T_{p1}^{au} (origin p, destination 1)
3	Value of T_{p2}^{au} (origin p, destination 2)
⋮	
q+1	Value of T_{pq}^{au} (origin p, destination 2)
⋮	
NCENT+1	Value of $T_{p,NCENT}^{au}$ (origin p , destination NCENT)

File name : GPQAU File number: 633

Contents : Predicted demand matrix of person trips by automobile

Type : Intermediate file

Symbol : g^{au}_{pq}

Created by : 6.60 RASS

Input to : 5.41 ODTRAN

reports

File format : Sequential

Binary

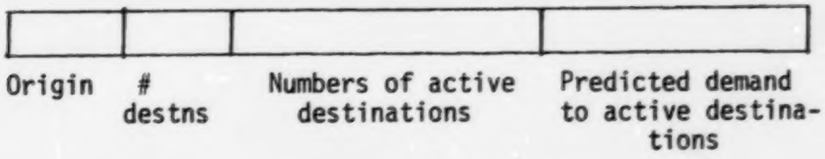
Record length (variable) = $2 * NACT_p + 2$; $NACT_p$ = number active destinations, origin p . For IBM, length = $6 * NACT_p + 4$ bytes.

The file contains 1 record of identification followed by 1 record for each origin p , p = 1, NCENT.

Record formats : a) Identification record

Word	Contents
1	Run id day
2	time
3	Number of iterations
4-13	Text (40 chars, 4 chars per word)

b) Origin records, one record for each origin p, p = 1, NCENT in following format.


Origin
# destns
Numbers of active destinations
Predicted demand to active destinations

See also "Packed OD matrix file format"

File name : DC0K File number: 701

Contents : Demand function coefficients

Type : Intermediate file

Symbol : $k0_p$, $k2_p$, $k3_p$, $k4_p$

Created by : User specified (from calibration programs)

Used by : 6.60 RASS ($k2_p$)
5.50 TASS
5.30 CALKTR

File format : Sequential

Coded

Record length = 24 characters

Record p contains values $k0_p$, $k2_p$, $k3_p$, $k4_p$

File contains 1 record for each origin p ; p= 1, NCENT.

Record format :

Cols	Fortran format	Contents
1-6	F6.3	$k0_p$ for origin p
7-12	F6.3	$k2_p$ "
13-18	F6.3	$k3_p$ "
19-24	F6.3	$k4_p$ "

Use : 1. In RASS, use is identical to that of file 611 DC0KAU.

File name : DC0K01 File number: 702

Contents : Component of demand function which is independent of transit and automobile impedances

Type : Work file

Symbol : $K01_{pq}$

Created by : 7.10 DCOS01

Input to: : 6.25 CALCK

5.30 CALKTR

File format : Sequential

Binary

Record length = NCENT words

Record p contains the values $K01_{pq}$; q = 1, NCENT

The file contains 1 record for each p ; p = 1, NCENT.

Record format :

Word	Contents
1	Value of $K01_{p1}$ (origin p , destination 1)
2	Value of $K01_{p2}$ (origin p , destination 2)
q	Value of $K01_{pq}$ (origin p , destination q)
NCENT	Value of $K01_{p,NCENT}$ (origin p , destination NCENT)

FILE NAME : PSPEC FILE NUMBER: 800

CONTENTS : Plot parameters.

TYPE : Direct entry input file.

INPUT TO : 8.22 DINAUN
8.42 DINAUV
8.62 DINTRN
8.82 DINTRV

FILE FORMAT : Sequential

Number of records - from 3 to 6 records depending on plot program being used.

RECORD FORMAT: Records required by all plots

Record 1 M1,M2,N1,N2.

Record 2 I1,I2,J1,J2.

These 8 integer words define area to be plotted as indicated in Fig. 800.1; i.e. they are external node numbers of nodes defining the rectangles as indicated. If no rectangle internal to plot area is to be omitted from plot, I1 is set to zero.

Record 3 SCALE, ANGROT, ANGNOR.

All variables are of type real

SCALE	Desired scale of plot (ft./in.).
ANGROT	Angle of rotation (counterclockwise sense) of plot axes of coordinates in data bank.
ANGNOR	Angle of north direction indicator to x-axis in counterclockwise sense, data bank coordinate system.

Record 4 Required for transit and vehicle volume plot only.

VMAX upper bound for link volume (type real)

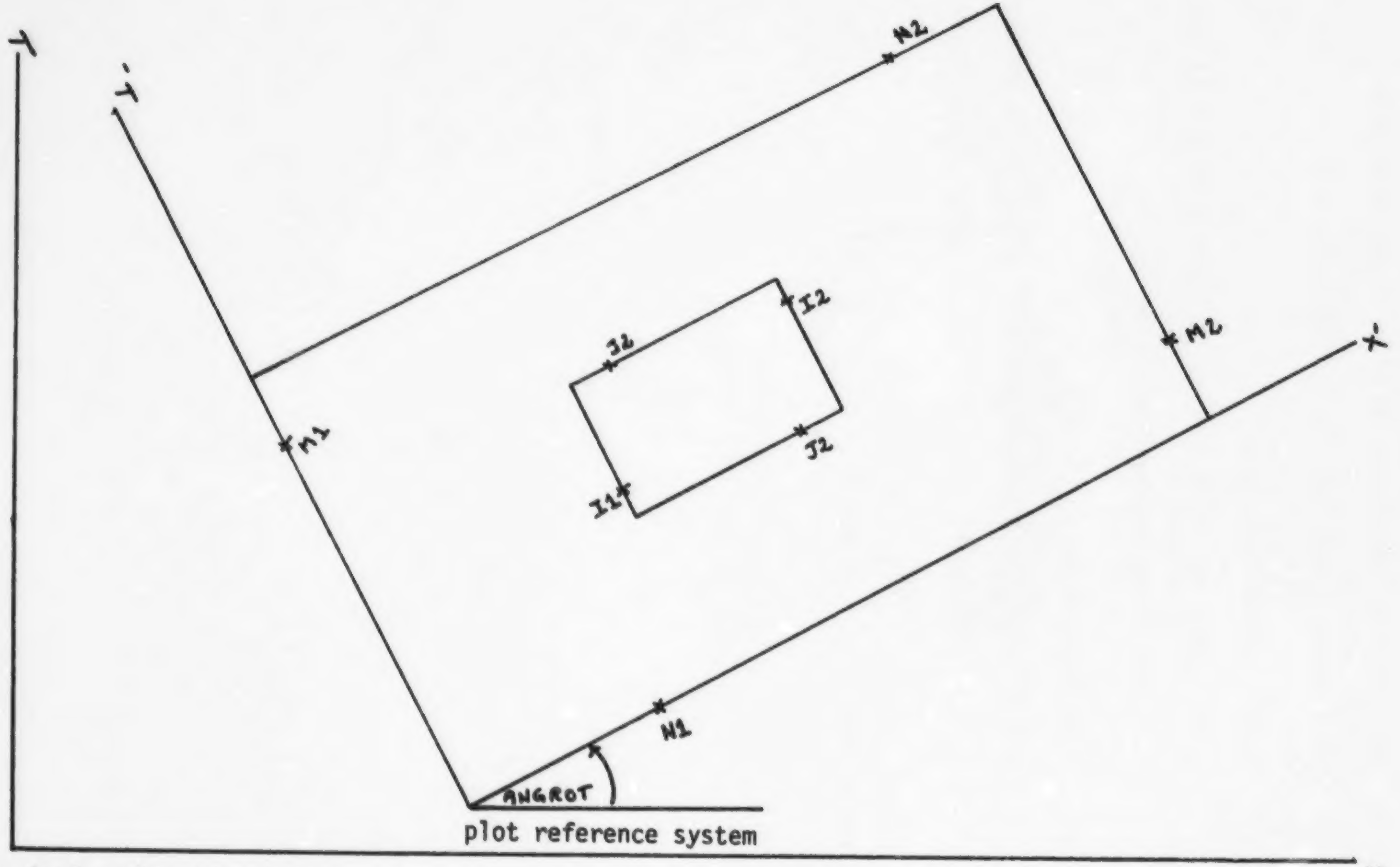
Record 5 Transit plots only.

LIND number of transit lines included in plot. If all lines are included, LIND is zero.

Record 6 Transit plots only.

LINENO Array with LIND elements of transit line numbers (external) to be included in plot. Not required if LIND is zero.

USE : Required by each plot program.


Y
Y'
X'
X
M1
M2
N1
N2
I1
I2
J1
J2
ANGROT
plot reference system

Data bank reference system

Fig. 800.1 M1,M2,N1,N2 are external node numbers delimiting plot area.

I1,I2,J1,J2 are external node numbers defining region internal to plot area to be omitted from plot.

FILE NAME : PTITDT FILE NUMBER: 801

CONTENTS : Plot title

TYPE : Direct entry input file

INPUT TO : 8.01 PTITLE

FILE FORMAT : Sequential

Number of records = 1

RECORD FORMAT: Record 1 contains NCHAR, TIT

NCHAR = Number of characters in title (≤60)

TIT = Alphanumeric variable of up to 60 characters describing current plot.

USE : Required by each plot program.

CF 11/5/78

Appendix D: Cross reference module and file index

This appendix contains the following indexes:

- modules listed in numerical order with brief description and reference to module description, if any
- modules listed in alphabetical order with module number
- files listed in numerical order with brief description
- files listed in alphabetical order with file number.

```
                              M O D U L E   L I S T
                           N U M E R I C A L   O R D E R

MODULE   MODULE   DESCRIPTION  - - - - - - - - - - - - - -  PURPOSE  - - - - - - - - - - - - - -
NUMBER    NAME     IN MODULE

          RBO                  REPLACE BLANK CHARACTER BY ZERO
          CKEND                CHECK FOR -END-
          RWIND                REWRITE DATABANK INDEX ON DISK
          BLOCD                CDC ONLY,BLOCK DATA ROUTINE
 0.01     GMEM        0.01     MEMORY ASSIGNMENT
 0.02     BLIREI
 0.02     BLIRE                READ A BLOCK FROM DATA BANK
 0.02     BECR
 0.04     BRECR                REURITE BLOCK IN DATA BANK
 0.07     CKBLNK               CHECK INPUT FIELD FOR BLANKS
 0.08     REEL                 CONVERT CHARACTER STRING TO A REAL NUMBER
 0.09     PRT                  PRINT INPUT DATA,IF NOT ALREADY PRINTED
 0.10     LETTER               CHECK IF GIVEN CHARACTER IS A LETTER
 0.11     SKBLK                SKIP BLANKS
 0.11     SKSEP                SKIP SEPARATOR (BLANK OR COMMA)
 0.11     SKNBLK               SKIP NON-BLANK
 0.11     SKNSEP               SKIP NON-SEPARATOR (OTHER THAN BLANK OR COMMA)
 0.15     CNV                  CONVERT REAL NUMBER TO CHARACTERS
 0.21     RDBIN                READ BINARY FILE
 0.22     PACKOD               WRITE MATRIX RECORD IN PACKED OD MATRIX FORMAT
 0.23     UPACOD               READ MATRIX AND UNPACK
 0.30     TURLIN               DETERMINE IF ROAD LINK TYPE SPECIFIES TURNING LINK
 1.00     EMSUP       1.00     EMME SUPERVISOR,COMMAND PROCESSOR (MAIN PROGRAM)
 1.01     ANCOM                ANALYSE DATA BANK SUBCOMMANDS ETC
 1.02     MODBAN               DATABANK SUBCOMMAND AND DATA SUPERVISOR
 1.09     BLOCK                BLOCK DATA SUBPROGRAM FOR SUPERVISOR
 1.10     INISUP      1.00     INITIALIZE EMME SYSTEM
 1.11     RDCOM       1.00     READ NEXT COMMAND,SUBCOMMAND,DATA FROM INPUT (COMFIL)
 1.12     ANWORD      1.12     ANALYZE NEXT WORD IN COMMAND,SUBCOMMAND
 1.13     SKNSPC               SKIP NON SEPARATOR (NOT A BLANK OR COMMA)
 1.13     SKACH                SKIP OCCURRENCES OF A GIVEN CHARACTER
 1.13     SKSEPC               SKIP SEPARATOR (BLANK OR COMMA)
 1.13     SKNACH               SKIP CHARACTERS OTHER THAN GIVEN CHARACTER
 1.15     DECDEB      1.15     DECODE DEBUG COMMAND OPTION
 1.16     DECEND               DECODE END COMMAND OPTION
 1.30     DECCOM      1.30     DECODE COMMAND
 1.40     DECTIT      1.40     SAVE TEXT FROM TITLE COMMAND
 1.60     DECASS      1.60     DECODE ASSIGN COMMAND OPTIONS AND SUBCOMMANDS
 1.62     DECPAR      1.62     DECODE PARAMETERS SUBCOMMAND
 1.63     GTWORD      1.63     GET NEXT WORD IN COMMAND,SUBCOMMAND.CONVERT IF NUMERC
 1.70     DECREP      1.70     DECODE REPORT OPTIONS AND SUBCOMMANDS
 2.01     BSORTF               BUBBLE SORT WITH FLAGS, 2 VECTORS
 2.01     BSORTG               BUBBLE SORT 1 VECTOR
 2.02     QSORT                SORT A VECTOR TOGETHER WITH A DATA VECTOR (QUICKSORT)
 2.02     QSORT1               SORT A VECTOR (QUICKSORT)
```

MODULE LIST
NUMERICAL ORDER

MODULE NUMBER	MODULE NAME	DESCRIPTION IN MODULE	PURPOSE
2.03	QESP		CHECK FOR REQUIRED SPACE IN A DATABANK BLOCK
2.04	QSPF		WRITE A NEW DATABANK BLOCK ON DISK
2.05	NXBLOC		GET NEXT BLOCK FROM DATABANK
2.06	QCEN		SEARCH SORTED VECTOR FOR GIVEN ITEM
2.09	HEADER		RECOGNIZE HEADER JUST READ
2.10	BCEN	2.10	RESTORE BLOCKS OF CENTROIDS
2.11	TCEN	2.11	SORT CENTROIDS BY EXTERNAL NUMBER
2.12	ACEN	2.12	ADD CENTROID TO LIST OF CENTROIDS
2.13	BMINI	2.13	RESTORE BLOCKS OF TURNING NODES
2.14	AMINI	2.14	ADD ENTRY TO NODE CORRESPONDENCE TABLE
2.15	IMINI	2.15	ASSIGN INTERNAL NUMBERS TO CORRESPONDENCE TABLE NODES
2.16	SRLO		CONSTRUCT LIST OF TURNING NODES
2.17	BRL	2.17	RESTORE ROAD LINKS
2.18	SRL	2.18	SEPARATE ROAD LINKS INTO SECTIONS,RENUMBER AND SORT
2.19	SRL1		PUT ROAD LINK IN APPROPIATE RECORD (IN CORE)
2.20	SRL2		SORT A TYPE OF ROAD LINK
2.21	ARL	2.21	ADD ROAD LINK
2.22	BTRO	2.22	RESTORE BLOCK OF TRANSIT-ONLY NODES
2.23	ITRO	2.23	ASSIGN INTERNAL NUMBERS TO TRANSIT-ONLY NODES
2.24	BLINE	2.24	RESTORE BLOCK OF TRANSIT LINES
2.25	ALINE	2.25	ADD TRANSIT LINE
2.26	ILINE	2.26	ASSIGN INTERNAL NUMBERS TO NODES ON TRANSIT LINES
2.27	BTL	2.27	RESTORE BLOCKS OF TRANSIT LINKS, DETERMINE TYPE
2.28	BTL1		RESTORE BLOCK OF GIVEN TYPE TRANSIT LINK
2.29	ALNK	2.29	ADD TRANSIT LINK, DETERMINE TYPE
2.30	ALNK1		ADD TRANSIT LINK OF SPECIFIC TYPE
2.31	ITL	2.31	ASSIGN INTERNAL NUMBERS TO TRANSIT LINK NODES
2.32	ITL2		RENUMBER TRANSIT LINKS OF GIVEN TYPE
2.33	ITL1		DETERMINE NODE TYPE OF GIVEN NODE
2.34	BMAT	2.34	RESTORE A GIVEN MATRIX
2.35	BMAT1		RESTORE AN OD MATRIX
2.36	BMAT2		RESTORE PART OF AN OD MATRIX
2.37	AODMAT	2.37	ADD OD MATRIX DATA
2.38	AMAT		ADD DATA TO A MATRIX
2.39	IMAT	2.39	ASSIGN INTERNAL NUMBERS TO CENTROIDS IN GIVEN MATRIX
2.40	IMAT1		ASSIGN INTERNAL NUMBERS TO CENTROIDS IN A MATRIX
2.41	IMAT2		RENUMBER PART OF MATRIX,ORIGIN OR DESTN
2.42	RCEN	2.42	READ CENTROIDS FROM INPUT (COMFIL)
2.43	RMINI	2.43	READ CORRESPONDENCE TABLE FROM INPUT (COMFIL)
2.44	RRL	2.44	READ ROAD LINKS FROM INPUT (COMFIL)
2.45	RLINE	2.45	READ TRANSIT LINE DATA FROM INPUT (COMFIL)
2.46	RTL	2.46	READ TRANSIT LINKS FROM INPUT (COMFIL)
2.47	RMAT	2.47	READ MATRIX DATA FROM INPUT (COMFIL)
2.48	DRL	2.48	DELETE A ROAD LINK
2.49	MRL	2.49	MODIFY A ROAD LINK

MODULE LIST
NUMERICAL ORDER

MODULE NUMBER	MODULE NAME	DESCRIPTION IN MODULE	- - - - - - - - - - - - - - PURPOSE - - - - - - - - - - - - - -
2.50	DCEN	2.50	DELETE A CENTROID
2.51	MCEN	2.51	MODIFY A CENTROID
2.52	DMINI	2.52	DELETE AN ENTRY IN CORRESPONDENCE TABLE
2.53	MMINI	2.53	MODIFY CORRESPONDENCE TABLE ENTRY
2.54	DLINE	2.54	DELETE A TRANSIT LINE
2.55	MLINE	2.55	MODIFY A TRANSIT LINE
2.56	DLNK	2.56	DELETE A TRANSIT LINK, DETERMINE TYPE
2.57	DLNK1		DELETE A GIVEN TYPE OF TRANSIT LINK
2.58	MLNK	2.58	MODIFY A TRANSIT LINK
2.59	MLNK1		MODIFY A GIVEN TYPE OF TRANSIT LINK
2.60	KEEPND	2.60	SAVE (KEEP) NODE NUMERS IN DATABANK
2.61	KPND1		SAVE (KEEP) NUMBERS FOR A GIVEN NODE TYPE IN DATABANK
2.62	B		CLEAN UP DATA BANK
2.63	RZONEG	2.63	READ ZONE GROUPS FROM INPUT (COMFIL)
2.64	RDFFTC	2.64	READ DEMAND FUNCTION TIME COEFFS FROM INPUT (COMFIL)
2.65	ADFTTC	2.65	ADD DEMAND FUNCTION TRAVEL TIME COEFFICIENTS
2.66	DDFTTC	2.66	DELETE DEMAND FUNCTION TRAVEL TIME COEFF
2.67	MDFTTC	2.67	MODIFY DEMAND FUNCTION TRAVEL TIME COEFFICIENTS
2.68	BDFTTC	2.68	CLEAN UP RECORD OF DEMAND FUNCTION TRAVEL TIME COEFF
2.69	AZONEG	2.69	ADD ZONE GROUP
2.70	DZONEG	2.70	DELETE A (PART OF) ZONE GROUP
2.71	BZONEG	2.71	CLEAN UP ZONE GROUPS
2.72	RSE1	2.72	READ SOCIONOMIC DATA FROM INPUT (COMFIL) FORMAT 1
2.73	ASE	2.73	ADD SOCIONOMIC COEEFICIENTS OR VARIABLES,FORMAT 1
2.74	MSE	2.74	MODIFY SOCIONOMIC VAR OR COEFFS,FORMAT 1+2(TRANSFORM)
2.75	DSE	2.75	DELETE SOCIONOMIC VAR OR COEFF,FORMAT 1 OR 2 TRANSFRM
2.76	AKEY	2.76	ADD KEYS OF SOCIO-ECONOMIC COEFFICIENTS OR VARIABLES
2.77	DKEY	2.77	DELETE KEYS OF SOCIONOMIC COEFF OR VARIABLES
2.78	RSE2	2.78	READ SOCIONOMIC DATA FROM INPUT (COMFIL) FORMAT 2
2.79	DSE2	2.79	DELETE SOCIONOMIC VAR OR COEFF,FORMAT 2
2.80	MSE2	2.80	MODIFY SOCIONOMIC VAR OR COEFFS,FORMAT 2
2.81	ASE2	2.81	ADD SOCIONOMIC COEEFICIENTS OR VARIABLES,FORMAT 2
2.82	ISE	2.82	EXPAND AND RENUMBER SOCIONOMIC VARIABLES AND COEFFS
2.84	BSE	2.84	CLEAN UP RECORDS OF SOCIONOMIC COEFF OR VARIABLES
2.85	ISE1		FIND ZONE GROUP IN CORE OR IN DATABANK
2.86	IDFTTC	2.86	RENUMBER AND VALIDATE DEMAND FUNCTION TIME COEFFS
2.87	OCEN	2.87	READ CENTROID FROM CURRENT SCENARIO
2.88	OMINI	2.88	READ CORRESPONDENCE TABLE FROM CURRENT SCENARIO
2.89	OLINE	2.89	READ TRANSIT LINES FROM CURRENT SCENARIO
2.90	ORL	2.90	READ ROAD LINKS FROM CURRENT SCENARIO
2.91	OTL	2.91	READ TRANSIT LINKS FROM CURRENT SCENARIO
2.92	OMAT	2.92	READ A MATRIX FROM CURRENT SCENARIO
2.93	OZONEG	2.93	READ ZONE GROUPS FROM CURRENT SCENARIO
2.94	OSE1	2.94	READ SOCIONOMIC VARS + COEFFS FROM CURRENT SCENARIO
2.95	OSE2	2.95	READ SOCIONOMIC DATA FROM CURRENT SCENARIO,FORMAT 2

MODULE LIST
NUMERICAL ORDER

MODULE NUMBER	MODULE NAME	DESCRIPTION IN MODULE	- - - - - - - - - - - - - - PURPOSE - - - - - - - - - - - - -
2.96	ODFFTC	2.96	READ DEMAND FUNCTION TIME COEFF FROM CURRENT SCENARIO
2.97	CR59		INITIALIZE RECORD 59 (SOCIONOMIC DATA)
2.98	SCENO	2.98	DEFINE A SCENARIO AND PREPARE RECORD FOR ACCUMULATION
2.99	ACCUM	2.99	ACCUMULATE CARD IMAGE OF A SCENARIO
2.100	MODMAT	2.100	MODIFY AN OD MATRIX
2.101	MMAT		MODIFY A GIVEN MATRIX
2.102	DMAT		DELETE ENTRIES IN A GIVEN MATRIX
2.102	DODMAT	2.102	DELETE ELEMENTS OF AN O-D MATRIX
2.104	RGEO	2.104	READ NODE COORDINATES FROM INPUT (COMFIL)
2.105	OGEO	2.105	READ NODE COORDINATES FROM CURRENT SCENARIO
2.106	AGEO	2.106	INSERT COORDINATES OF A NODE
2.107	DGEO	2.107	DELETE COORDINATES OF A NODE
2.108	MGEO	2.108	MODIFY COORDINATES OF A NODE
2.109	BGEO	2.109	CLEAN UP RECORD OF NODE COORDINATES
2.110	IGEO	2.110	RENUMBER AND SORT NODE COORDINATES
2.111	SCALE		READ COORDINATE SCALE FROM INPUT AND PLACE IN DATABNK
2.112	RTPM	2.112	READ TRANSIT PATH MULTIPLIERS FROM INPUT (COMFIL)
2.113	ATPM	2.113	ADD TRANSIT PATH MULTIPLIERS
2.114	DTPM	2.114	DELETE TRANSIT PATH MULTIPLIERS
2.115	MTPM	2.115	MODIFY TRANSIT PATH MULTIPLIERS
2.116	BTPM	2.116	CLEAN UP RECORD OF TRANSIT PATH MULTIPLIERS
2.117	ITPM	2.117	RENUMBER AND VALIDATE TRANSIT PATH MULTIPLIERS
2.118	OTPM	2.118	READ TRANSIT PATH MULTIPLIERS FROM CURRENT SCENARIO
2.119	IJ		SORT AND RENUMBER RECORDS IN DATABANK
2.120	REMS		REMOVE EFFECT OF USED SCENARIOS FROM ACTIVE DATA
3.01	RDLINE		READ DATA FOR NEXT TRANSIT LINE FROM DATABANK
3.02	GTLINE		GET TRANSIT LINE CHARACTERISTICS
3.03	RDCOR		READ NEXT ENTRY IN CORRESPONDENCE TABLE FROM DATABANK
3.04	RDTL		READ NEXT TRANSIT LINK OF GIVEN TYPE FROM DATABANK
3.05	PSEGRL		SET DATABANK TO BEGINNING OF TRANSIT EGRESS LINKS
3.05	RDACCL		READ TRANSIT ACCESS LINKS FROM INPUT (COMFIL)
3.05	RDTLT		READ TRANSIT-ONLY LINKS FROM INPUT (COMFIL)
3.05	PSTLT		SET DATABANK TO BEGINNING OF TRANSIT-ONLY LINKS
3.05	RDEGRL		READ TRANSIT EGRESS LINKS FROM INPUT (COMFIL)
3.05	PSACCL		SET DATABANK TO BEGINNING OF TRANSIT ACCESS LINKS
3.11	RDODM		READ DATA FOR NEXT ORIGIN FROM A GIVEN MATRIX
3.12	RDMEST		READ MODAL SPLIT FOR NEXT ORIGIN FROM DATABANK
3.12	RDOD		READ OD MATRIX DATA FOR NEXT ORIGIN FROM DATABANK
3.12	RDOCC		READ AUTO OCCUPANCY DATA FOR NEXT ORIGIN FROM DATABNK
3.12	RDODEX		READ VEHICLE ADJUSTMENT DATA FOR NEXT ORIGIN
3.13	PSOD		SET DATABANK TO BEGINNING OF AN OD MATRIX
3.13	PSOCC		SET DATABANK TO BEGINNING OF AUTO OCCUPANCY MATRIX
3.13	PSMEST		SET DATABANK TO BEGINNING OF MODAL SPLIT MATRIX
3.13	PSODEX		SET DATABANK TO BEGINNING OF VEHICLE ADJUST MATRIX
3.14	GTNODE		GET EXTERNAL NODE NUMBER AND COORDINATES OF NODE

MODULE LIST
NUMERICAL ORDER

MODULE NUMBER	MODULE NAME	DESCRIPTION IN MODULE	PURPOSE
3.15	PSLINK		SET DATABANK TO BEGINNING OF SECTION OF ROAD LINKS
3.16	RDLINK		READ NEXT ROAD LINK FROM DATABANK
3.17	RDRL		READ NEXT ROAD LINK FROM GIVEN SECTION FROM DATABANK
3.18	RLINKU		READ USER DEFINED DATA FOR NEXT ROAD LINK
3.19	RLINKV		READ USER DATA FOR NEXT ROAD LINK IN GIVEN SECTION
3.20	PTNODE		GET INTERNAL NUMBER FOR GIVEN EXTERNAL NODE NUMBER
3.21	OPBAN	3.21	OPEN DATABANK,IF NOT OPEN, AND INITIALIZE
3.22	GETND	3.22	READ EXTERNAL NODE NUMBERS FROM DATABANK
3.23	RDDCOC		READ NEXT SET OF SOCIONOMIC COEFFS FROM DATABANK
3.23	RDDCOC		READ NEXT SET OF SOCIONOMIC VARIABLES
3.24	RDDCOK		READ DEMAND FUNCTION TIME COEFFS FOR NEXT ORIGIN
3.25	PSDCOK		SET DATABANK TO BEGINNING OF DEMAND FUNCTION COEFFS
3.26	PURGE		PROCESS DATABANK PURGE SUBCOMMAND
3.33	GTNODV		GET EXTERNAL NODE NUMBERS FOR GIVEN INTERNAL NUMBERS
3.34	RDDVC		READ NEXT SET OF SOCIONOMIC VARIABLES OR COEFFS
3.36	PSDVAR		POSITION DATABANK TO READ SOCIONOMIC VARIABLES
3.36	PSDCOC		POSITION DATABANK TO READ SOCIONOMIC COEFFS
3.38	RECY		
3.39	PUR		PURGE DATABANK RECORD WHICH IS A MATRIX
3.40	MUCCA	3.40	GET NEXT CARD IMAGE FROM CURRENT SCENARIO
3.41	SCEN1	3.41	CHECK IF SCENARIO DEFINED AND PUT IN STACK
3.42	SCEN2	3.42	INITIALIZE NEXT SCENARIO
3.43	PCEN		PRINT CENTROIDS
3.44	PLINE		PRINT TRANSIT LINES
3.45	PTL		PRINT A GIVEN TYPE OF TRANSIT LINKS,DETERMINE TYPE
3.46	PTL1		PRINT A GIVEN TYPE OF TRANSIT LINKS
3.47	PMINI		PRINT CORRESPONDENCE TABLE
3.48	PMAT		PRINT A MATRIX
3.49	PRL		PRINT ROAD LINKS
3.50	PDFTTC		PRINT DEMAND FUNCTION TRAVEL TIME COEFFS
3.51	PZONEG		PRINT ZONE GROUPS
3.52	PSE		PRINT SOCIONOMIC VARIABLES OR COEFFICIENTS
3.53	PSCEN		PRINT SCENARIO
3.54	SCENF		ACCUMULATE END CARD OF A SCENARIO
3.55	SCEN3		ERASE THE NAME OF A SCENARIO
3.56	PRINT		PROCESS DATABANK PRINT SUBCOMMAND
3.57	NAMEOD		GET NAMES OF 3 OD MATRICES
3.58	GTSCAL		GET SCALE OF COORDINATES
3.59	PGEO		PRINT NODE COORDINATES
3.60	RDPMUW		READ TRANSIT PATH MULTIPLIERS FOR NEXT ORIGIN
3.61	PSPMUW		SET DATABANK TO BEGINNING OF TRANSIT PATH MULTIPLIERS
3.62	PTPM		PRINT TRANSIT PATH MULTIPLIERS
3.63	UCEN		PUNCH CENTROIDS
3.64	UGEO		PUNCH NODE COORDINATES
3.65	ULINE		PUNCH TRANSIT LINES

MODULE LIST
NUMERICAL ORDER

MODULE NUMBER	MODULE NAME	DESCRIPTION IN MODULE	- - - - - - - - - - - - - - PURPOSE - - - - - - - - - - - - - -
3.66	UMAT		PUNCH A MATRIX
3.67	UMINI		PUNCH CORRESPONDENCE TABLE
3.68	URL		PUNCH ROAD LINKS
3.69	USCEN		PUNCH A SCENARIO
3.70	UTLI		PUNCH TRANSIT LINKS OF GIVEN TYPE
3.71	UTL		PUNCH TRANSIT LINKS,DETERMINE TYPE
3.72	UZCNEG		PUNCH ZONE GROUPS
3.74	UTPM		PUNCH TRANSIT PATH MULTIPLIERS
3.75	PUNCH		PROCESS DATABANK PUNCH SUBCOMMAND
3.76	USE		PUNCH SOCIONOMIC VARIABLES OR COEFICIENTS
3.77	PSLINE		SET DATABANK TO BEGINNING OF TRANSIT LINES
3.78	PSCOR		SET DATABANK TO BEGINNING OF CORRESPONDENCE TABLE
4.00	ASSPRO	4.00	CONTROL ASSIGNMENT PROCEDURE ACCORDING TO OPTIONS
4.01	PRTASS		PRINT ASSIGNMENT PROCEDURE STATISTICS
4.15	RDLTV2	4.31	READ TIME AND VOLUME ON ROAD LINKS (SECTION 2)
4.18	BUSFRQ	4.18	CALCULATE NUMBER OF TRANSIT VEHICLES ON ROAD LINKS
4.18	UPTLFR	4.18	ENTRY POINT IN BUSFRQ TO UPDATE FILE TLFREQ
4.21	TIMPED	4.21	CALCULATE TRANSIT IMPEDANCES
4.21	MODIMP	4.21	CALCULATE AND COMPARE TRANSIT IMPEDANCES
4.31	TIMCHK	4.31	CALCULATE NEW TRANSIT LINK TIMES AND COMPARE
4.32	CHANGE	4.31	MODIFY TRANSIT LINK TIME IF DIFFERENCE IS SIGNIFICANT
4.41	BUSTIM	4.31	DETERMINE TIME ON TRANSIT-ONLY LINK
4.42	BUSCAR	4.31	BUS TIME BY MILE AS A FUNCTION OF CAR TIME BY MILE
4.43	BUSROD	4.31	CALCULATE BUS TIME ON A ROAD LINK
4.61	CAPCHK	4.61	ADJUST LINE CAPACITY (HEADWAYS) TO PASSENGER VOLUMES
5.00	RTNET		READ FILE TNET
5.01	WTNET		WRITE FILE TNET
5.02	DISTAN		CALCULATE DISTANCE BETWEEN TWO NODES FROM COORDINATES
5.03	TDISCO	5.15	DETERMINE TRANSIT LINK LENGTHS FROM COORDINATES
5.09	LINOUT	5.12	LIST TRANSIT LINES PASSING AT EACH NODE
5.10	GENER	5.10	GENERATE TRANSIT NETWORK
5.11	LISTLI		LIST TRANSIT LINES WITH CHARACTERISTICS
5.12	BLDNET	5.12	BUILD TRANSIT NETWORK FROM LINE DESCRIPTIONS
5.13	TIMOUT		PRINT TRANSIT LINK TIMES AND DISTANCES
5.14	EXPNET	5.14	CONSTRUCT EXPANDED TRANSIT NETWORK
5.15	TTIME	5.15	DEFINE INITIAL TRANSIT LINK TIMES
5.16	EQUIVA	5.14	DEFINE A NEW LINK IN EXPANDED FORWARD ITINERARY
5.17	EQUIVR	5.14	DEFINE A NEW LINK IN EXPANDED REVERSE ITINERARY
5.18	SEARCH	5.14	LOOK FOR A GIVEN LINK IN SECTION 2 OF FILE RLINK
5.19	WARNIN	5.14	PRINT OUT A MESSAGE
5.20	TPATH	4.21	CALCULATE TRANSIT SHORTEST PATHS
5.21	ACCESS	5.20	LABEL ALL NODES ACCESSIBLE BY A SINGLE LINE
5.22	FORLAB	5.20	LABEL NODES ON FORWARD ITINERARY
5.23	REVLAB	5.20	LABEL NODES ON REVERSE ITINERARY
5.24	LABEL	5.20	UPDATE NODE LABEL AND HEAP

```
                              M O D U L E   L I S T
                            N U M E R I C A L   O R D E R

MODULE  MODULE  DESCRIPTION  - - - - - - - - - - - - - - PURPOSE - - - - - - - - - - - - - -
NUMBER   NAME    IN MODULE

 5.25   DELETE     5.20      DELETE NODE ON TOP OF THE HEAP AND REORDER
 5.26   INSER      5.20      INSERT NEW NODE IN THE HEAP
 5.27   MODIF      5.20      MODIFY TIME LABEL OF A NODE IN THE HEAP AND REORDER
 5.30   CALKTR     5.30      CALC DEMAND FUNC COMPONENT INDEPENDANT OF TRANSIT
 5.31   ACCESX     5.20      VERSION OF 5.21 ACCESS FOR COMMON LINES ALGORITHM
 5.32   FORLAX     5.20      VERSION OF 5.22 FORLAB FOR COMMON LINES ALGORITHM
 5.33   REVLAX     5.20      VERSION OF 5.23 REVLAB FOR COMMON LINES ALGORITHM
 5.41   ODTRAN     5.41      CALCULATE CURRANT DEMAND FOR FIXED DEM TRAN ASSIGN
 5.50   TASS       5.50      TRANSIT ASSIGNMENT PROCEDURE
 5.55   TDEM       5.50      GET ARRAY OF TRANSIT DEMAND FROM A GIVEN ORIGIN
 5.56   LOASEC     5.50      CUMULATE PASSENGERS ON A LINE SECTION
 5.60   FLOOUT     5.50      PRINT TRANSIT VOLUMES AND SAVE SOLUTION
 5.61   GCODE                DETERMINE A CODE TO INDICATE A SET OF COMMON LINES
 6.00   RINIT                INITIALIZE ROAD TRAFFIC DEMAND
 6.10   AUTRIP     6.10      FIND AUTO EQUIV OF TOTAL TRIPS(AND INIT DEMAND)
 6.15   INITDM     6.15      FIND INITIAL AUTO DEMAND
 6.20   TRAFIC               PERFORM ROAD TRAFFIC ASSIGNMENT,FIXED OR VAR DEMAND
 6.25   CALCK      6.25      CALC DEMAND FUNC COMPONENT INDEPENDENT OF AUTO TIME
 6.41   NETWRK     6.41      READ IN ROAD LINKS AND CONSTRUCT NETWORK TABLES
 6.50   BUSIN      6.50      READ AND STORE BUS FREQUENCY ON EACH ROAD LINK
 6.60   RASS       6.60      ROAD TRAFFIC ASSIGNMENT PROCEDURE
 6.61   LOADTR     6.60      ACCUMULATE FLOW ON LINKS FROM DEMAND FROM GIVEN ORIGN
 6.62   MINTRE     6.60      CONSTRUCT SHORTEST PATH TREE FROM GIVEN ORIGIN
 6.63   OBJFN      6.60      CALCULATE OBJECTIVE FUNCTION VALUES
 6.64   OPSTEP     6.60      DETERMINE LAMBDA TO MINIMIZE OBJECTIVE FUNCTION
 6.66   MOES       6.60      CALCULATE TOTAL VEHICLE MILES,HOURS
 6.71   USECST     6.60      CALCULATE USER COST*TRAVEL TIME FOR GIVEN ROAD LINK
 6.72   TLCST      6.60      CALCULATE INTEGRAL OF USER COST FOR GIVEN ROAD LINK
 6.73   DEMCST     6.60      CALCULATE INVERSE OF DEMAND FUNCTION FOR AN OD PAIR
 6.74   TDCST      6.60      CALCULATE INTEGRAL OF INVERSE OF DEMAND FUNCTION
 6.75   LINKIN     6.41      READ USER LINK DATA. SAMPLE VERSION
 6.76   FINDLC     6.60      GET ROAD LINK CHARACTERISTICS. SAMPLE VERSION
 6.77   FINDCC     6.60      GET DELAY CURVE CHARACTERISTICS. SAMPLE VERSION
 6.80   RASSAV     6.80      SAVE ROAD TRAFFIC ASSIGNMENT SOLUTION VALUES
 6.92   PRTGPQ     6.92      PRINT PREDICTED AUTOMOBILE DEMAND REPORT
 6.94   PRTLNK     6.94      PRINT PREDICTED ROAD LINK VOLUME AND TRAVEL TIME
 6.95   RDRLTV     6.95      READ ROAD TRAFFIC SOLUTION,FILE RLTIVL
 6.96   RTRIPD     6.96      PRINT ROAD NETWORK TRIP TIME DISTRIBUTION
 7.10   DCOS01     7.10      CALCULATE DEMAND FUNC COMPONENT INDEPENDENT OF IMPED
 8.01   PTITLE               PLOTS TITLE AND DATE
 8.02   HEADNG               PRINTS SCALE AND DIRECTION INDICATOR ON PLOT
 8.03   COMPL                CHECKS IF ALL LINKS EXITING A GIVEN NODE HAVE BEEN PLOTTED
 8.04   ANGLE                DETERMINES ANGLE OF A LINE TO THE X-AXIS
 8.05   VOL                  DRAWS VOLUME LINES FOR A GIVEN LINK
 8.06   VOLNUM               PLOTS VOLUME NUMBER FOR GIVEN LINK
```

MODULE LIST
NUMERICAL ORDER

MODULE NUMBER	MODULE NAME	DESCRIPTION IN MODULE	- - - - - - - - - - - - - - PURPOSE - - - - - - - - - - - - - -
8.07	BAR		PLOTS VOLUME SCALE
8.09	BLKDTA		BLOCK DATA SUBPROGRAM - DEFINES PLOT PARAMETERS
8.10	DPARAM		CALCULATES BEGINNING AND END POINTS FOR PLOTTING LINK MEDIAN
8.11	DASH		PLOTS DASHED LINE BETWEEN TWO GIVEN POINTS
8.12	CONFGA		DETERMINES STRUCTURE OF VOLUME SEGMENT - CONFIGURATION "A"
8.13	CONFGB		DETERMINES STRUCTURE OF VOLUME SEGMENT - CONFIGURATION "B"
8.14	CONFGC		DETERMINES STRUCTURE OF VOLUME SEGMENT - CONFIGURATION "C"
8.15	CONFGD		DETERMINES STRUCTURE OF VOLUME SEGMENT - CONFIGURATION "D"
8.21	PLOTVN		MAIN PROGRAM - VEHICLE NETWORK PLOT
8.22	DINAUN		READS USER SUPPLIED INPUT DATA - VEHICLE NETWORK PLOT
8.23	NETPLT		DETERMINES SUBNETWORK TO BE PLOTTED - VEHICLE NETWORK PLOT
8.24	ANALGO		ORDERS PLOTTING OF NETWORK LINKS - VEHICLE NETWORK PLOT
8.25	ERRDST		COMPARES CALCULATED AND MEASURED VEHICLE LINK LENGTHS
8.26	ANNODE		PLOTS NODE NUMBER - VEHICLE NETWORK PLOT
8.27	LINKPL		PLOTS LINK MEDIAN - VEHICLE NETWORK PLOT
8.41	AUVOLP		MAIN PROGRAM - VEHICLE VOLUME PLOT
8.42	DINAUV		READS USER SUPPLIED INPUT DATA - VEHICLE VOLUME PLOT
8.43	TABLE		READS IN LINK VOLUMES FROM SOLUTION FILE
8.44	VOLPLT		DETERMINES LINKS IN PLOT AREA
8.45	AVALGO		ORDERS PLOTTING OF LINKS - VEHICLE VOLUME PLOT
8.46	AVNODE		PLOTS NODE NUMBER - VEHICLE VOLUME PLOT
8.47	AVLINE		DETERMINE CATEGORY OF VOLUME SEGMENT - VEHICLE VOLUME PLOT
8.48	CNUM		PLOT PREDICTED VOLUME ON CENTROID CONNECTOR
8.61	TRNETP		MAIN PROGRAM - TRANSIT NETWORK PLOT
8.62	DINTRN		READS USER SUPPLIED INPUT DATA - TRANSIT NETWORK
8.63	ROUTE		DETERMINES BUS LINES USING EACH LINK
8.64	TABTN		ACCUMLATES LINKS FOR EACH BUS LINE INTO A LINK TABLE
8.65	INSERT		INSERTS A TRANSIT LINK INTO LINK TABLE
8.66	TRANV		APPENDS BUS LINE NUMBER ONTO LINK IN LINK TABLE
8.67	SORT		CONVERTS INTERNAL LINE NUMBER TO EXTERNAL
8.68	TRPLOT		DETERMINES NUMBER OF LINKS EXITING NODE - TRANSIT PLOT
8.69	TNALGO		ORDERS PLOTTING OF LINKS - TRANSIT NETWORK
8.70	FLECHE		PLOTS TRANSIT LINK
8.71	CHECTN		CHECKS IF TRANSIT LINK IS TWO-WAY - TRANSIT NETWORK PLOT
8.72	TNNODE		PLOTS NODE NUMBER - TRANSIT NETWORK PLOT
8.73	LINKTN		PLOTS TRANSIT LINK MEDIAN AND LINE NUMBERS
8.74	TNACES		PLOTS CENTROID CONNECTORS - TRANSIT NETWORK PLOT
8.81	TRVOLP		MAIN PROGRAM - TRANSIT VOLUME PLOT
8.82	DINTRV		READS INPUT DATA - TRANSIT VOLUME PLOT
8.83	VOLACC		ACCUMULATES TRANSIT LINK VOLUMES
8.84	ORDER		CONSTRUCTS TRANSIT LINK TABLE
8.85	ACC1		INSERTS TRANSIT LINK IN LINK TABLE
8.86	ACC2		INCREASE VOLUME ON TRANSIT
8.87	TRANFR		DEFINE TRANSIT LINK WITH LOWER NODE NUMBERS - VOLUME PLOT
8.90	PLTRVL		ACCESSES PLOT ROUTINES - TRANSIT VOLUME PLOT

```
                              M O D U L E   L I S T
                            N U M E R I C A L   O R D E R

MODULE  MODULE  DESCRIPTION  - - - - - - - - - - - - - PURPOSE - - - - - - - - - - - - - -
NUMBER   NAME    IN MODULE

 8.91   TVALGO               ORDERS PLOTTING OF LINKS - TRANSIT VOLUME PLOT
 8.92   CENLIN               PLOTS LINK MEDIAN
 8.93   CHECTV               CHECKS IF LINK MEDIAN HAS BEEN PLOTTED
 8.94   TVNODE               PLOTS NODE NUMBER - TRANSIT VOLUME PLOT
 8.95   TVACES               PLOT ACCESS AND EGRESS LINKS - TRANSIT VOLUME PLOT
 8.96   TVLINE               DETERMINE CATEGORY OF VOLUME SEGMENT - TRANSIT VOLUME PLOT
 9.01   TTIT                 PRINT REPORT TITLES
 9.21   CORRD2               PROCESS CORRIDOR SPECS,FLAG ROAD LINKS IN CORRIDOR
```

```
                              M O D U L E   L I S T
                         A L P H A B E T I C A L   O R D E R

MODULE  MODULE  DESCRIPTION  - - - - - - - - - - - - - - PURPOSE - - - - - - - - - - - - - -
NUMBER   NAME    IN MODULE

 5.21   ACCESS     5.20      LABEL ALL NODES ACCESSIBLE BY A SINGLE LINE
 5.31   ACCESX     5.20      VERSION OF 5.21 ACCESS FOR COMMON LINES ALGORITHM
 2.99   ACCUM      2.99      ACCUMULATE CARD IMAGE OF A SCENARIO
 6.85   ACC1                 INSERTS TRANSIT LINK IN LINK TABLE
 8.86   ACC2                 INCREASE VOLUME ON TRANSIT
 2.12   ACEN       2.12      ADD CENTROID TO LIST OF CENTROIDS
 2.65   ADFTTC     2.65      ADD DEMAND FUNCTION TRAVEL TIME COEFFICIENTS
 2.106  AGEO       2.106     INSERT COORDINATES OF A NODE
 2.76   AKEY       2.76      ADD KEYS OF SOCIO-ECONOMIC COEFFICIENTS OR VARIABLES
 2.25   ALINE      2.25      ADD TRANSIT LINE
 2.29   ALNK       2.29      ADD TRANSIT LINK, DETERMINE TYPE
 2.30   ALNK1                ADD TRANSIT LINK OF SPECIFIC TYPE
 2.38   AMAT                 ADD DATA TO A MATRIX
 2.14   AMINI      2.14      ADD ENTRY TO NODE CORRESPONDENCE TABLE
 8.24   ANALGO               ORDERS PLOTTING OF NETWORK LINKS - VEHICLE NETWORK PLOT
 1.01   ANCOM                ANALYSE DATA BANK SUBCOMMANDS ETC
 8.04   ANGLE                DETERMINES ANGLE OF A LINE TO THE X-AXIS
 8.26   ANNODE               PLOTS NODE NUMBER - VEHICLE NETWORK PLOT
 1.12   ANWORD     1.12      ANALYZE NEXT WORD IN COMMAND,SUBCOMMAND
 2.37   AODMAT     2.37      ADD OD MATRIX DATA
 2.21   ARL        2.21      ADD ROAD LINK
 2.73   ASE        2.73      ADD SOCIONOMIC COEEFICIENTS OR VARIABLES,FORMAT 1
 2.81   ASE2       2.81      ADD SOCIONOMIC COEEFICIENTS OR VARIABLES,FORMAT 2
 4.00   ASSPRO     4.00      CONTROL ASSIGNMENT PROCEDURE ACCORDING TO OPTIONS
 2.113  ATPM       2.113     ADD TRANSIT PATH MULTIPLIERS
 6.10   AUTRIP     6.10      FIND AUTO EQUIV OF TOTAL TRIPS(AND INIT DEMAND)
 8.41   AUVOLP               MAIN PROGRAM - VEHICLE VOLUME PLOT
 8.45   AVALGO               ORDERS PLOTTING OF LINKS - VEHICLE VOLUME PLOT
 8.47   AVLINE               DETERMINE CATEGORY OF VOLUME SEGMENT - VEHICLE VOLUME PLOT
 8.46   AVNODE               PLOTS NODE NUMBER - VEHICLE VOLUME PLOT
 2.69   AZONEG     2.69      ADD ZONE GROUP
 2.62   B                    CLEAN UP DATA BANK
 8.07   BAR                  PLOTS VOLUME SCALE
 2.10   BCEN       2.10      RESTORE BLOCKS OF CENTROIDS
 2.68   BDFTTC     2.68      CLEAN UP RECORD OF DEMAND FUNCTION TRAVEL TIME COEFF
 0.02   BECR
 2.109  BGEO       2.109     CLEAN UP RECORD OF NODE COORDINATES
 5.12   BLDNET     5.12      BUILD TRANSIT NETWORK FROM LINE DESCRIPTIONS
 2.24   BLINE      2.24      RESTORE BLOCK OF TRANSIT LINES
 0.02   BLIRE                READ A BLOCK FROM DATA BANK
 0.02   BLIRE1
 8.09   BLKDTA               BLOCK DATA SUBPROGRAM - DEFINES PLOT PARAMETERS
  .     BLOCD                CDC ONLY,BLOCK DATA ROUTINE
 1.09   BLOCK                BLOCK DATA SUBPROGRAM FOR SUPERVISOR
 2.34   BMAT       2.34      RESTORE A GIVEN MATRIX
```

MODULE LIST
ALPHABETICAL ORDER

MODULE NUMBER	MODULE NAME	DESCRIPTION IN MODULE	PURPOSE
2.35	BMAT1		RESTORE AN OD MATRIX
2.36	BMAT2		RESTORE PART OF AN OD MATRIX
2.13	BMINI	2.13	RESTORE BLOCKS OF TURNING NODES
0.04	BRECR		REURITE BLOCK IN DATA BANK
2.17	BRL	2.17	RESTORE ROAD LINKS
2.84	BSE	2.84	CLEAN UP RECORDS OF SOCIONOMIC COEFF OR VARIABLES
2.01	BSORTF		BUBBLE SORT WITH FLAGS, 2 VECTORS
2.01	BSORTG		BUBBLE SORT 1 VECTOR
2.27	BTL	2.27	RESTORE BLOCKS OF TRANSIT LINKS, DETERMINE TYPE
2.28	BTL1		RESTORE BLOCK OF GIVEN TYPE TRANSIT LINK
2.116	BTPM	2.116	CLEAN UP RECORD OF TRANSIT PATH MULTIPLIERS
2.22	BTRO	2.22	RESTORE BLOCK OF TRANSIT-ONLY NODES
4.42	BUSCAR	4.31	BUS TIME BY MILE AS A FUNCTION OF CAR TIME BY MILE
4.18	BUSFRQ	4.18	CALCULATE NUMBER OF TRANSIT VEHICLES ON ROAD LINKS
6.50	BUSIN	6.50	READ AND STORE BUS FREQUENCY ON EACH ROAD LINK
4.43	BUSROD	4.31	CALCULATE BUS TIME ON A ROAD LINK
4.41	BUSTIM	4.31	DETERMINE TIME ON TRANSIT-ONLY LINK
2.71	BZONEG	2.71	CLEAN UP ZONE GROUPS
6.25	CALCK	6.25	CALC DEMAND FUNC COMPONENT INDEPENDENT OF AUTO TIME
5.30	CALKTR	5.30	CALC DEMAND FUNC COMPONENT INDEPENDANT OF TRANSIT
4.61	CAPCHK	4.61	ADJUST LINE CAPACITY (HEADWAYS) TO PASSENGER VOLUMES
8.92	CENLIN		PLOTS LINK MEDIAN
4.32	CHANGE	4.31	MODIFY TRANSIT LINK TIME IF DIFFERENCE IS SIGNIFICANT
8.71	CHECTN		CHECKS IF TRANSIT LINK IS TWO-WAY - TRANSIT NETWORK PLOT
8.93	CHECTV		CHECKS IF LINK MEDIAN HAS BEEN PLOTTED
0.07	CKBLNK		CHECK INPUT FIELD FOR BLANKS
	CKEND		CHECK FOR -END-
8.48	CNUM		PLOT PREDICTED VOLUME ON CENTROID CONNECTOR
0.15	CNV		CONVERT REAL NUMBER TO CHARACTERS
8.03	COMPL		CHECKS IF ALL LINKS EXITING A GIVEN NODE HAVE BEEN PLOTTED
8.12	CONFGA		DETERMINES STRUCTURE OF VOLUME SEGMENT - CONFIGURATION "A"
8.13	CONFGB		DETERMINES STRUCTURE OF VOLUME SEGMENT - CONFIGURATION "B"
8.14	CONFGC		DETERMINES STRUCTURE OF VOLUME SEGMENT - CONFIGURATION "C"
8.15	CONFGD		DETERMINES STRUCTURE OF VOLUME SEGMENT - CONFIGURATION "D"
9.21	CORRD2		PROCESS CORRIDOR SPECS.FLAG ROAD LINKS IN CORRIDOR
2.97	CR59		INITIALIZE RECORD 59 (SOCIONOMIC DATA)
8.11	DASH		PLOTS DASHED LINE BETWEEN TWO GIVEN POINTS
2.50	DCEN	2.50	DELETE A CENTROID
7.10	DCOS01	7.10	CALCULATE DEMAND FUNC COMPONENT INDEPENDENT OF IMPED
2.66	DDFTTC	2.66	DELETE DEMAND FUNCTION TRAVEL TIME COEFF
1.60	DECASS	1.60	DECODE ASSIGN COMMAND OPTIONS AND SUBCOMMANDS
1.30	DECCOM	1.30	DECODE COMMAND
1.15	DECDEB	1.15	DECODE DEBUG COMMAND OPTION
1.16	DECEND		DECODE END COMMAND OPTION
1.62	DECPAR	1.62	DECODE PARAMETERS SUBCOMMAND

MODULE LIST
ALPHABETICAL ORDER

MODULE NUMBER	MODULE NAME	DESCRIPTION IN MODULE	- - - - - - - - - - - - - PURPOSE - - - - - - - - - - - - -
1.70	DECREP	1.70	DECODE REPORT OPTIONS AND SUBCOMMANDS
1.40	DECTIT	1.40	SAVE TEXT FROM TITLE COMMAND
5.25	DELETE	5.20	DELETE NODE ON TOP OF THE HEAP AND REORDER
6.73	DEMCST	6.60	CALCULATE INVERSE OF DEMAND FUNCTION FOR AN OD PAIR
2.107	DGEO	2.107	DELETE COORDINATES OF A NODE
8.22	DINAUN		READS USER SUPPLIED INPUT DATA - VEHICLE NETWORK PLOT
8.42	DINAUV		READS USER SUPPLIED INPUT DATA - VEHICLE VOLUME PLOT
8.62	DINTRN		READS USER SUPPLIED INPUT DATA - TRANSIT NETWORK
8.82	DINTRV		READS INPUT DATA - TRANSIT VOLUME PLOT
5.02	DISTAN		CALCULATE DISTANCE BETWEEN TWO NODES FROM COORDINATES
2.77	DKEY	2.77	DELETE KEYS OF SOCIONOMIC COEFF OR VARIABLES
2.54	DLINE	2.54	DELETE A TRANSIT LINE
2.56	DLNK	2.56	DELETE A TRANSIT LINK, DETERMINE TYPE
2.57	DLNK1		DELETE A GIVEN TYPE OF TRANSIT LINK
2.102	DMAT		DELETE ENTRIES IN A GIVEN MATRIX
2.52	DMINI	2.52	DELETE AN ENTRY IN CORRESPONDENCE TABLE
2.102	DODMAT	2.102	DELETE ELEMENTS OF AN O-D MATRIX
8.10	DPARAM		CALCULATES BEGINNING AND END POINTS FOR PLOTTING LINK MEDIAN
2.48	DRL	2.48	DELETE A ROAD LINK
2.75	DSE	2.75	DELETE SOCIONOMIC VAR OR COEFF,FORMAT 1 OR 2 TRANSFRM
2.79	DSE2	2.79	DELETE SOCIONOMIC VAR OR COEFF,FORMAT 2
2.114	DTPM	2.114	DELETE TRANSIT PATH MULTIPLIERS
2.70	DZONEG	2.70	DELETE A (PART OF) ZONE GROUP
1.00	EMSUP	1.00	EMME SUPERVISOR,COMMAND PROCESSOR (MAIN PROGRAM)
5.16	EQUIVA	5.14	DEFINE A NEW LINK IN EXPANDED FORWARD ITINERARY
5.17	EQUIVR	5.14	DEFINE A NEW LINK IN EXPANDED REVERSE ITINERARY
8.25	ERRDST		COMPARES CALCULATED AND MEASURED VEHICLE LINK LENGTHS
5.14	EXPNET	5.14	CONSTRUCT EXPANDED TRANSIT NETWORK
6.77	FINDCC	6.60	GET DELAY CURVE CHARACTERISTICS. SAMPLE VERSION
6.76	FINDLC	6.60	GET ROAD LINK CHARACTERISTICS. SAMPLE VERSION
8.70	FLECHE		PLOTS TRANSIT LINK
5.60	FLOOUT	5.50	PRINT TRANSIT VOLUMES AND SAVE SOLUTION
5.22	FORLAB	5.20	LABEL NODES ON FORWARD ITINERARY
5.32	FORLAX	5.20	VERSION OF 5.22 FORLAB FOR COMMON LINES ALGORITHM
5.61	GCODE		DETERMINE A CODE TO INDICATE A SET OF COMMON LINES
5.10	GENER	5.10	GENERATE TRANSIT NETWORK
3.22	GETND	3.22	READ EXTERNAL NODE NUMBERS FROM DATABANK
0.01	GMEM	0.01	MEMORY ASSIGNMENT
3.02	GTLINE		GET TRANSIT LINE CHARACTERISTICS
3.14	GTNODE		GET EXTERNAL NODE NUMBER AND COORDINATES OF NODE
3.33	GTNODV		GET EXTERNAL NODE NUMBERS FOR GIVEN INTERNAL NUMBERS
3.58	GTSCAL		GET SCALE OF COORDINATES
1.63	GTWORD	1.63	GET NEXT WORD IN COMMAND,SUBCOMMAND.CONVERT IF NUMERC
2.09	HEADER		RECOGNIZE HEADER JUST READ
8.02	HEADNG		PRINTS SCALE AND DIRECTION INDICATOR ON PLOT

MODULE LIST
ALPHABETICAL ORDER

MODULE NUMBER	MODULE NAME	DESCRIPTION IN MODULE	- - - - - - - - - - - - - - PURPOSE - - - - - - - - - - - - - -
2.86	IDFTTC	2.86	RENUMBER AND VALIDATE DEMAND FUNCTION TIME COEFFS
2.110	IGEO	2.110	RENUMBER AND SORT NODE COORDINATES
2.119	IJ		SORT AND RENUMBER RECORDS IN DATABANK
2.26	ILINE	2.26	ASSIGN INTERNAL NUMBERS TO NODES ON TRANSIT LINES
2.39	IMAT	2.39	ASSIGN INTERNAL NUMBERS TO CENTROIDS IN GIVEN MATRIX
2.40	IMAT1		ASSIGN INTERNAL NUMBERS TO CENTROIDS IN A MATRIX
2.41	IMAT2		RENUMBER PART OF MATRIX,ORIGIN OR DESTN
2.15	IMINI	2.15	ASSIGN INTERNAL NUMBERS TO CORRESPONDENCE TABLE NODES
1.10	INISUP	1.00	INITIALIZE EMME SYSTEM
6.15	INITDM	6.15	FIND INITIAL AUTO DEMAND
5.26	INSER	5.20	INSERT NEW NODE IN THE HEAP
8.65	INSERT		INSERTS A TRANSIT LINK INTO LINK TABLE
2.82	ISE	2.82	EXPAND AND RENUMBER SOCIONOMIC VARIABLES AND COEFFS
2.85	ISE1		FIND ZONE GROUP IN CORE OR IN DATABANK
2.31	ITL	2.31	ASSIGN INTERNAL NUMBERS TO TRANSIT LINK NODES
2.33	ITL1		DETERMINE NODE TYPE OF GIVEN NODE
2.32	ITL2		RENUMBER TRANSIT LINKS OF GIVEN TYPE
2.117	ITPM	2.117	RENUMBER AND VALIDATE TRANSIT PATH MULTIPLIERS
2.23	ITRO	2.23	ASSIGN INTERNAL NUMBERS TO TRANSIT-ONLY NODES
2.60	KEEPND	2.60	SAVE (KEEP) NODE NUMERS IN DATABANK
2.61	KPND1		SAVE (KEEP) NUMBERS FOR A GIVEN NODE TYPE IN DATABANK
5.24	LABEL	5.20	UPDATE NODE LABEL AND HEAP
0.10	LETTER		CHECK IF GIVEN CHARACTER IS A LETTER
6.75	LINKIN	6.41	READ USER LINK DATA. SAMPLE VERSION
8.27	LINKPL		PLOTS LINK MEDIAN - VEHICLE NETWORK PLOT
8.73	LINKTN		PLOTS TRANSIT LINK MEDIAN AND LINE NUMBERS
5.09	LINOUT	5.12	LIST TRANSIT LINES PASSING AT EACH NODE
5.11	LISTLI		LIST TRANSIT LINES WITH CHARACTERISTICS
6.61	LOADTR	6.60	ACCUMULATE FLOW ON LINKS FROM DEMAND FROM GIVEN ORIGN
5.56	LOASEC	5.50	CUMULATE PASSENGERS ON A LINE SECTION
2.51	MCEN	2.51	MODIFY A CENTROID
2.67	MDFTTC	2.67	MODIFY DEMAND FUNCTION TRAVEL TIME COEFFICIENTS
2.108	MGEO	2.108	MODIFY COORDINATES OF A NODE
6.62	MINTRE	6.60	CONSTRUCT SHORTEST PATH TREE FROM GIVEN ORIGIN
2.55	MLINE	2.55	MODIFY A TRANSIT LINE
2.58	MLNK	2.58	MODIFY A TRANSIT LINK
2.59	MLNK1		MODIFY A GIVEN TYPE OF TRANSIT LINK
2.101	MMAT		MODIFY A GIVEN MATRIX
2.53	MMINI	2.53	MODIFY CORRESPONDENCE TABLE ENTRY
1.02	MODBAN		DATABANK SUBCOMMAND AND DATA SUPERVISOR
5.27	MODIF	5.20	MODIFY TIME LABEL OF A NODE IN THE HEAP AND REORDER
4.21	MODIMP	4.21	CALCULATE AND COMPARE TRANSIT IMPEDANCES
2.100	MODMAT	2.100	MODIFY AN OD MATRIX
6.68	MOES	6.60	CALCULATE TOTAL VEHICLE MILES,HOURS
2.49	MRL	2.49	MODIFY A ROAD LINK

```
                          M O D U L E   L I S T
                   A L P H A B E T I C A L   O R D E R

MODULE  MODULE  DESCRIPTION  - - - - - - - - - - - - - - PURPOSE - - - - - - - - - - - - - -
NUMBER   NAME    IN MODULE

 2.74   MSE       2.74       MODIFY SOCIONOMIC VAR OR COEFFS,FORMAT 1+2(TRANSFORM)
 2.80   MSE2      2.80       MODIFY SOCIONOMIC VAR OR COEFFS,FORMAT 2
 2.115  MTPM      2.115      MODIFY TRANSIT PATH MULTIPLIERS
 3.40   MUCCA     3.40       GET NEXT CARD IMAGE FROM CURRENT SCENARIO
 3.57   NAMEOD               GET NAMES OF 3 OD MATRICES
 8.23   NETPLT               DETERMINES SUBNETWORK TO BE PLOTTED - VEHICLE NETWORK PLOT
 6.41   NETWRK    6.41       READ IN ROAD LINKS AND CONSTRUCT NETWORK TABLES
 2.05   NXBLOC               GET NEXT BLOCK FROM DATABANK
 6.63   OBJFN     6.60       CALCULATE OBJECTIVE FUNCTION VALUES
 2.87   OCEN      2.87       READ CENTROID FROM CURRENT SCENARIO
 2.96   ODFFTC    2.96       READ DEMAND FUNCTION TIME COEFF FROM CURRENT SCENARIO
 5.41   ODTRAN    5.41       CALCULATE CURRANT DEMAND FOR FIXED DEM TRAN ASSIGN
 2.105  OGEO      2.105      READ NODE COORDINATES FROM CURRENT SCENARIO
 2.89   OLINE     2.89       READ TRANSIT LINES FROM CURRENT SCENARIO
 2.92   OMAT      2.92       READ A MATRIX FROM CURRENT SCENARIO
 2.88   OMINI     2.88       READ CORRESPONDENCE TABLE FROM CURRENT SCENARIO
 3.21   OPBAN     3.21       OPEN DATABANK,IF NOT OPEN, AND INITIALIZE
 6.64   OPSTEP    6.60       DETERMINE LAMBDA TO MINIMIZE OBJECTIVE FUNCTION
 8.84   ORDER                CONSTRUCTS TRANSIT LINK TABLE
 2.90   ORL       2.90       READ ROAD LINKS FROM CURRENT SCENARIO
 2.94   OSE1      2.94       READ SOCIONOMIC VARS + COEFFS FROM CURRENT SCENARIO
 2.95   OSE2      2.95       READ SOCIONOMIC DATA FROM CURRENT SCENARIO,FORMAT 2
 2.91   OTL       2.91       READ TRANSIT LINKS FROM CURRENT SCENARIO
 2.118  OTPM      2.118      READ TRANSIT PATH MULTIPLIERS FROM CURRENT SCENARIO
 2.93   OZONEG    2.93       READ ZONE GROUPS FROM CURRENT SCENARIO
 0.22   PACKOD               WRITE MATRIX RECORD IN PACKED OD MATRIX FORMAT
 3.43   PCEN                 PRINT CENTROIDS
 3.50   PDFTTC               PRINT DEMAND FUNCTION TRAVEL TIME COEFFS
 3.59   PGEO                 PRINT NODE COORDINATES
 3.44   PLINE                PRINT TRANSIT LINES
 8.21   PLOTVN               MAIN PROGRAM - VEHICLE NETWORK PLOT
 8.90   PLTRVL               ACCESSES PLOT ROUTINES - TRANSIT VOLUME PLOT
 3.48   PMAT                 PRINT A MATRIX
 3.47   PMINI                PRINT CORRESPONDENCE TABLE
 3.56   PRINT                PROCESS DATABANK PRINT SUBCOMMAND
 3.49   PRL                  PRINT ROAD LINKS
 0.09   PRT                  PRINT INPUT DATA,IF NOT ALREADY PRINTED
 4.01   PRTASS               PRINT ASSIGNMENT PROCEDURE STATISTICS
 6.92   PRTGPQ    6.92       PRINT PREDICTED AUTOMOBILE DEMAND REPORT
 6.94   PRTLNK    6.94       PRINT PREDICTED ROAD LINK VOLUME AND TRAVEL TIME
 3.05   PSACCL               SET DATABANK TO BEGINNING OF TRANSIT ACCESS LINKS
 3.53   PSCEN                PRINT SCENARIO
 3.78   PSCOR                SET DATABANK TO BEGINNING OF CORRESPONDENCE TABLE
 3.36   PSDCOC               POSITION DATABANK TO READ SOCIONOMIC COEFFS
 3.25   PSDCOK               SET DATABANK TO BEGINNING OF DEMAND FUNCTION COEFFS
```

MODULE LIST
ALPHABETICAL ORDER

MODULE NUMBER	MODULE NAME	DESCRIPTION IN MODULE	- - - - - - - - - - - - - - PURPOSE - - - - - - - - - - - - - -
3.36	PSDVAR		POSITION DATABANK TO READ SOCIONOMIC VARIABLES
3.52	PSE		PRINT SOCIONOMIC VARIABLES OR COEFFICIENTS
3.05	PSEGRL		SET DATABANK TO BEGINNING OF TRANSIT EGRESS LINKS
3.77	PSLINE		SET DATABANK TO BEGINNING OF TRANSIT LINES
3.15	PSLINK		SET DATABANK TO BEGINNING OF SECTION OF ROAD LINKS
3.13	PSMEST		SET DATABANK TO BEGINNING OF MODAL SPLIT MATRIX
3.13	PSOCC		SET DATABANK TO BEGINNING OF AUTO OCCUPANCY MATRIX
3.13	PSOD		SET DATABANK TO BEGINNING OF AN OD MATRIX
3.13	PSODEX		SET DATABANK TO BEGINNING OF VEHICLE ADJUST MATRIX
3.61	PSPMUW		SET DATABANK TO BEGINNING OF TRANSIT PATH MULTIPLIERS
3.05	PSTLT		SET DATABANK TO BEGINNING OF TRANSIT-ONLY LINKS
8.01	PTITLE		PLOTS TITLE AND DATE
3.45	PTL		PRINT A GIVEN TYPE OF TRANSIT LINKS,DETERMINE TYPE
3.46	PTL1		PRINT A GIVEN TYPE OF TRANSIT LINKS
3.20	PTNODE		GET INTERNAL NUMBER FOR GIVEN EXTERNAL NODE NUMBER
3.62	PTPM		PRINT TRANSIT PATH MULTIPLIERS
3.75	PUNCH		PROCESS DATABANK PUNCH SUBCOMMAND
3.39	PUR		PURGE DATABANK RECORD WHICH IS A MATRIX
3.26	PURGE		PROCESS DATABANK PURGE SUBCOMMAND
3.51	PZONEG		PRINT ZONE GROUPS
2.06	QCEN		SEARCH SORTED VECTOR FOR GIVEN ITEM
2.03	QESP		CHECK FOR REQUIRED SPACE IN A DATABANK BLOCK
2.02	QSORT		SORT A VECTOR TOGETHER WITH A DATA VECTOR (QUICKSORT)
2.02	QSORT1		SORT A VECTOR (QUICKSORT)
2.04	QSPF		WRITE A NEW DATABANK BLOCK ON DISK
6.60	RASS	6.60	ROAD TRAFFIC ASSIGNMENT PROCEDURE
6.80	RASSAV	6.80	SAVE ROAD TRAFFIC ASSIGNMENT SOLUTION VALUES
	RBO		REPLACE BLANK CHARACTER BY ZERO
2.42	RCEN	2.42	READ CENTROIDS FROM INPUT (COMFIL)
3.05	RDACCL		READ TRANSIT ACCESS LINKS FROM INPUT (COMFIL)
0.21	RDBIN		READ BINARY FILE
1.11	RDCOM	1.00	READ NEXT COMMAND,SUBCOMMAND,DATA FROM INPUT (COMFIL)
3.03	RDCOR		READ NEXT ENTRY IN CORRESPONDENCE TABLE FROM DATABANK
3.23	RDDCOC		READ NEXT SET OF SOCIONOMIC COEFFS FROM DATABANK
3.23	RDDCOC		READ NEXT SET OF SOCIONOMIC VARIABLES
3.24	RDDCOK		READ DEMAND FUNCTION TIME COEFFS FOR NEXT ORIGIN
3.34	RDDVC		READ NEXT SET OF SOCIONOMIC VARIABLES OR COEFFS
3.05	RDEGRL		READ TRANSIT EGRESS LINKS FROM INPUT (COMFIL)
2.64	RDFFTC	2.64	READ DEMAND FUNCTION TIME COEFFS FROM INPUT (COMFIL)
3.01	RDLINE		READ DATA FOR NEXT TRANSIT LINE FROM DATABANK
3.16	RDLINK		READ NEXT ROAD LINK FROM DATABANK
4.15	RDLTV2	4.31	READ TIME AND VOLUME ON ROAD LINKS (SECTION 2)
3.12	RDMEST		READ MODAL SPLIT FOR NEXT ORIGIN FROM DATABANK
3.12	RDOCC		READ AUTO OCCUPANCY DATA FOR NEXT ORIGIN FROM DATABNK
3.12	RDOD		READ OD MATRIX DATA FOR NEXT ORIGIN FROM DATABANK

MODULE LIST
ALPHABETICAL ORDER

MODULE NUMBER	MODULE NAME	DESCRIPTION IN MODULE	PURPOSE
3.12	RDODEX		READ VEHICLE ADJUSTMENT DATA FOR NEXT ORIGIN
3.11	RDODM		READ DATA FOR NEXT ORIGIN FROM A GIVEN MATRIX
3.60	RDPMUW		READ TRANSIT PATH MULTIPLIERS FOR NEXT ORIGIN
3.17	RDRL		READ NEXT ROAD LINK FROM GIVEN SECTION FROM DATABANK
6.95	RDRLTV	6.95	READ ROAD TRAFFIC SOLUTION.FILE RLTIVL
3.04	RDTL		READ NEXT TRANSIT LINK OF GIVEN TYPE FROM DATABANK
3.05	RDTLT		READ TRANSIT-ONLY LINKS FROM INPUT (COMFIL)
3.38	RECV		
0.08	REEL		CONVERT CHARACTER STRING TO A REAL NUMBER
2.120	REMS		REMOVE EFFECT OF USED SCENARIOS FROM ACTIVE DATA
5.23	REVLAB	5.20	LABEL NODES ON REVERSE ITINERARY
5.33	REVLAX	5.20	VERSION OF 5.23 REVLAB FOR COMMON LINES ALGORITHM
2.104	RGEO	2.104	READ NODE COORDINATES FROM INPUT (COMFIL)
6.00	RINIT		INITIALIZE ROAD TRAFFIC DEMAND
2.45	RLINE	2.45	READ TRANSIT LINE DATA FROM INPUT (COMFIL)
3.18	RLINKU		READ USER DEFINED DATA FOR NEXT ROAD LINK
3.19	RLINKV		READ USER DATA FOR NEXT ROAD LINK IN GIVEN SECTION
2.47	RMAT	2.47	READ MATRIX DATA FROM INPUT (COMFIL)
2.43	RMINI	2.43	READ CORRESPONDENCE TABLE FROM INPUT (COMFIL)
8.63	ROUTE		DETERMINES BUS LINES USING EACH LINK
2.44	RRL	2.44	READ ROAD LINKS FROM INPUT (COMFIL)
2.72	RSE1	2.72	READ SOCIONOMIC DATA FROM INPUT (COMFIL) FORMAT 1
2.78	RSE2	2.78	READ SOCIONOMIC DATA FROM INPUT (COMFIL) FORMAT 2
2.46	RTL	2.46	READ TRANSIT LINKS FROM INPUT (COMFIL)
5.00	RTNET		READ FILE TNET
2.112	RTPM	2.112	READ TRANSIT PATH MULTIPLIERS FROM INPUT (COMFIL)
6.96	RTRIPD	6.96	PRINT ROAD NETWORK TRIP TIME DISTRIBUTION
	RWIND		REWRITE DATABANK INDEX ON DISK
2.63	RZONEG	2.63	READ ZONE GROUPS FROM INPUT (COMFIL)
2.111	SCALE		READ COORDINATE SCALE FROM INPUT AND PLACE IN DATABNK
3.54	SCENF		ACCUMULATE END CARD OF A SCENARIO
2.98	SCENO	2.98	DEFINE A SCENARIO AND PREPARE RECORD FOR ACCUMULATION
3.41	SCEN1	3.41	CHECK IF SCENARIO DEFINED AND PUT IN STACK
3.42	SCEN2	3.42	INITIALIZE NEXT SCENARIO
3.55	SCEN3		ERASE THE NAME OF A SCENARIO
5.18	SEARCH	5.14	LOOK FOR A GIVEN LINK IN SECTION 2 OF FILE RLINK
1.13	SKACH		SKIP OCCURRENCES OF A GIVEN CHARACTER
0.11	SKBLK		SKIP BLANKS
1.13	SKNACH		SKIP CHARACTERS OTHER THAN GIVEN CHARACTER
0.11	SKNBLK		SKIP NON-BLANK
0.11	SKNSEP		SKIP NON-SEPARATOR (OTHER THAN BLANK OR COMMA)
1.13	SKNSPC		SKIP NON SEPARATOR (NOT A BLANK OR COMMA)
0.11	SKSEP		SKIP SEPARATOR (BLANK OR COMMA)
1.13	SKSEPC		SKIP SEPARATOR (BLANK OR COMMA)
8.67	SORT		CONVERTS INTERNAL LINE NUMBER TO EXTERNAL

MODULE LIST
ALPHABETICAL ORDER

MODULE NUMBER	MODULE NAME	DESCRIPTION IN MODULE	- - - - - - - - - - - - - - PURPOSE - - - - - - - - - - - - - -
2.18	SRL	2.18	SEPARATE ROAD LINKS INTO SECTIONS,RENUMBER AND SORT
2.16	SRLO		CONSTRUCT LIST OF TURNING NODES
2.19	SRL1		PUT ROAD LINK IN APPROPIATE RECORD (IN CORE)
2.20	SRL2		SORT A TYPE OF ROAD LINK
8.43	TABLE		READS IN LINK VOLUMES FROM SOLUTION FILE
8.64	TABTN		ACCUMLATES LINKS FOR EACH BUS LINE INTO A LINK TABLE
5.50	TASS	5.50	TRANSIT ASSIGNMENT PROCEDURE
2.11	TCEN	2.11	SORT CENTROIDS BY EXTERNAL NUMBER
6.74	TDCST	6.60	CALCULATE INTEGRAL OF INVERSE OF DEMAND FUNCTION
5.55	TDEM	5.50	GET ARRAY OF TRANSIT DEMAND FROM A GIVEN ORIGIN
5.03	TDISCO	5.15	DETERMINE TRANSIT LINK LENGTHS FROM COORDINATES
4.31	TIMCHK	4.31	CALCULATE NEW TRANSIT LINK TIMES AND COMPARE
5.13	TIMOUT		PRINT TRANSIT LINK TIMES AND DISTANCES
4.21	TIMPED	4.21	CALCULATE TRANSIT IMPEDANCES
6.72	TLCST	6.60	CALCULATE INTEGRAL OF USER COST FOR GIVEN ROAD LINK
8.74	TNACES		PLOTS CENTROID CONNECTORS - TRANSIT NETWORK PLOT
8.69	TNALGO		ORDERS PLOTTING OF LINKS - TRANSIT NETWORK
8.72	TNNODE		PLOTS NODE NUMBER - TRANSIT NETWORK PLOT
5.20	TPATH	4.21	CALCULATE TRANSIT SHORTEST PATHS
6.20	TRAFIC		PERFORM ROAD TRAFFIC ASSIGNMENT,FIXED OR VAR DEMAND
8.87	TRANFR		DEFINE TRANSIT LINK WITH LOWER NODE NUMBERS - VOLUME PLOT
8.66	TRANV		APPENDS BUS LINE NUMBER ONTO LINK IN LINK TABLE
8.61	TRNETP		MAIN PROGRAM - TRANSIT NETWORK PLOT
8.68	TRPLOT		DETERMINES NUMBER OF LINKS EXITING NODE - TRANSIT PLOT
8.81	TRVOLP		MAIN PROGRAM - TRANSIT VOLUME PLOT
5.15	TTIME	5.15	DEFINE INITIAL TRANSIT LINK TIMES
9.01	TTIT		PRINT REPORT TITLES
0.30	TURLIN		DETERMINE IF ROAD LINK TYPE SPECIFIES TURNING LINK
8.95	TVACES		PLOT ACCESS AND EGRESS LINKS - TRANSIT VOLUME PLOT
8.91	TVALGO		ORDERS PLOTTING OF LINKS - TRANSIT VOLUME PLOT
8.96	TVLINE		DETERMINE CATEGORY OF VOLUME SEGMENT - TRANSIT VOLUME PLOT
8.94	TVNODE		PLOTS NODE NUMBER - TRANSIT VOLUME PLOT
3.63	UCEN		PUNCH CENTROIDS
3.64	UGEO		PUNCH NODE COORDINATES
3.65	ULINE		PUNCH TRANSIT LINES
3.66	UMAT		PUNCH A MATRIX
3.67	UMINI		PUNCH CORRESPONDENCE TABLE
0.23	UPACOD		READ MATRIX AND UNPACK
4.18	UPTLFR	4.18	ENTRY POINT IN BUSFRQ TO UPDATE FILE TLFREQ
3.68	URL		PUNCH ROAD LINKS
3.69	USCEN		PUNCH A SCENARIO
3.76	USE		PUNCH SOCIONOMIC VARIABLES OR COEFICIENTS
6.71	USECST	6.60	CALCULATE USER COST*TRAVEL TIME FOR GIVEN ROAD LINK
3.71	UTL		PUNCH TRANSIT LINKS,DETERMINE TYPE
3.70	UTL1		PUNCH TRANSIT LINKS OF GIVEN TYPE

```
                         M O D U L E   L I S T
                   A L P H A B E T I C A L   O R D E R

MODULE  MODULE  DESCRIPTION  - - - - - - - - - - - - - PURPOSE - - - - - - - - - - - - -
NUMBER   NAME    IN MODULE

 3.74   UTPM                 PUNCH TRANSIT PATH MULTIPLIERS
 3.72   UZONEG               PUNCH ZONE GROUPS
 8.05   VOL                  DRAWS VOLUME LINES FOR A GIVEN LINK
 8.83   VOLACC               ACCUMULATES TRANSIT LINK VOLUMES
 8.06   VOLNUM               PLOTS VOLUME NUMBER FOR GIVEN LINK
 8.44   VOLPLT               DETERMINES LINKS IN PLOT AREA
 5.19   WARNIN     5.14      PRINT OUT A MESSAGE
 5.01   WTNET                WRITE FILE TNET
```

EMME: File index (Numerical order)

Number	Name	Contents
200	NODES	Geocoded node list (internal)
201	LINES	Description of transit lines
202	NODCOR	Node correspondence table
203	ACCESS EGRESS	Transit {access {egress links
204	TLTIM	Transit link time, initial estimate
205	INITIM	Initial transit time, road links
211	RLINK	Road network link list
221	PERSPQ	OD Matrix (Persons) $\bar{g}_{pq}$
222	AUTOCC	Auto occupancy γ_{pq}
223	MODEST	Modal split estimate
224	VEHADJ	Vehicle adjustment matrix
231	DCOK	Demand function coefficients $k0_p$, $k1_p$, $k2_p$
233	DCOC	Demand function socio-economic coefficients
234	DVAR	Demand function socio-economic variables
250	PMUW	Transit path multipliers
302	CENT	List of centroids
304	NODEIN	Geocoded nodes as read from input
336	DBSTAT	Data bank status
337	STOPS	Nodes on transit lines
339	RLNKIN	Road links as read from input
354	NODEXT	Internal-external node numbers
355	DCOKIN	Demand function coefficients as read from input
357	ZGROUP	Zone group definitions

EMME: File index (Numerical order)

Number	Name	Contents
359	SOCIO	Socio-economic variables and coefficients
361	SCEN	Scenarios
362	PMUWIN	Transit path multipliers as read from input
501	TLLIST	Transit network link list
511	DCOKTR	Demand function component K_{pq}^{tr}
514	CORNET	Expanded transit link list
515	TNET	Transit network
518	TLFREQ	Bus frequency by link
521	TTREES	Shortest path on transit network
531	TZIMP	Transit impedance u_{pq}^{tr}
532	TZTIM	Interzone time T_{pq}^{tr}
541	GPQTR	Transit demand matrix g_{pq}^{tr}
550	TFLOW	Transit passenger flow by link
601	GBPQ	OD matrix, total vehicle equiv g_{pq}^{veh}
602	GPQRAN	OD matrix, auto g_{pq}^{car}
611	DCOKAU	Demand function component K_{pq}^{au}
621	DIRFIL	Direction d_{pq}
631	RLTIVL	Time, volume by link
632	RZTIM	Interzone time T_{pq}^{au}
633	GPQAU	Predicted demand g_{pq}^{au}
702	DCOK01	Demand function component $K01_{pq}$

EMME: File Index (Alphabetical order)

Number	Name	Contents
203	ACCESS EGRESS	Transit {access links egress
222	AUTOCC	Auto occupancy γ_{pq}
302	CENT	List of centroids
514	CORNET	Expanded transit link list
336	DBSTAT	Data bank status
233	DCOC	Demand function socio-economic coefficients
231	DCOK	Demand function coefficients $k0_p$, $k1_p$, $k2_p$
611	DCOKAU	Demand function component K^{au}_{pq}
702	DCOK01	Demand function component $K01_{pq}$
355	DCOKIN	Demand function coefficients as read from input
511	DCOKTR	Demand function component K^{tr}_{pq}
621	DIRFIL	Direction d_{pq}
234	DVAR	Demand function socio-economic variables
601	GBPQ	OD matrix, total vehicle equiv g^{veh}_{pq}
633	GPQAU	Predicted demand g^{au}_{pq}
602	GPQRAN	OD matrix, auto g^{car}_{pq}
541	GPQTR	Transit demand matrix g^{tr}_{pq}
205	INITIM	Initial transit time, road links
201	LINES	Description of transit lines
223	MODEST	Modal split estimate
202	NODCOR	Node correspondence table
304	NODEIN	Geocoded nodes as read from input
200	NODES	Geocoded node list (internal)

EMME: File Index (Alphabetical order)

Number	Name	Contents
354	NODEXT	Internal-external node numbers
221	PERSPQ	OD Matrix (persons)$\bar{g}_{pq}$
250	PMUW	Transit path multipliers
362	PMUWIN	Transit path multipliers as read from input
211	RLINK	Road network link list
339	RLNKIN	Road links as read from input
631	RLTIVL	Time, volume by link
632	RZTIM	Interzone time T^{au}_{pq}
361	SCEN	Scenarios
359	SOCIO	Socio-economic variables and coefficients
337	STOPS	Nodes on transit lines
550	TFLOW	Transit passenger flow by link
518	TLFREQ	Bus frequency by link
501	TLLIST	Transit network link list
204	TLTIM	Transit link time, initial estimate
515	TNET	Transit network
521	TTREES	Shortest path on transit network
531	TZIMP	Transit impedance u^{tr}_{pq}
532	TZTIM	Interzone time T^{tr}_{pq}
224	VEHADJ	Vehicle adjustment matrix
357	ZGROUP	Zone group definitions

Appendix E: Variable demand traffic assignment algorithm

(Pages 3-8 of "Procedures for Equilibrium Traffic Assignment with Elastic Demand", publication # 39, Centre de recherche sur les transports, Université de Montréal.

II Problem formulation

Consider a road network of n nodes and m oriented links. It is assumed that nodes are numbered from 1 to n; the first 1, 2, ..., n_0 nodes are origins and destinations. Let g_{pq} denote the travel demand between origin p and destination q, f_a^p the flow on link a coming from origin p, and f_a the total link flow. We assumed that the travel cost on link a varies according to a given monotone increasing function $C_a(f_a)$, and the travel demand g_{pq} varies according to a given monotone decreasing function $D_{pq}(u_{pq})$, where u_{pq} is the interzonal travel cost from p to q. Furthermore, we assume that the inverse function $G_{pq}(g_{pq})$ of $D_{pq}(u_{pq})$ exists and is integrable.

A traffic pattern is said to be in the equilibrium state when

1- it satisfies Wardrop's first principle (user optimized traffic)

2- the interzonal travel cost corresponding to the flows obtained with the current demand is equal to the cost given by the inverse of the demand function, when the demand is equal to the current demand.

Beckman *et al.* [15] have shown that the equilibrium traffic pattern is a solution of a minimization problem, which can be viewed as a convex cost multicommodity circulation flow problem. This problem may be defined as in the following:

$$\text{(P) minimize } Z\,(f_a, g_{pq}) = \sum_a \int_0^{f_a} C_a(x)dx - \sum_{pq} \int_{\ell_{pq}}^{g_{pq}} G_{pq}(x)dx \qquad (1)$$

subject to

$$\sum_{a \varepsilon W_i} f_a^p - \sum_{a \varepsilon V_i} f_a^p + \delta_{pi}\, g_{pi} = 0 \qquad (2)$$

$$i \neq p;\ i = 1,2,\ldots,n;\ p = 1,2,\ldots,n_0;$$

$$f_a^p > 0 \quad \text{all } a \ ; \ p = 1, 2, \ldots, n_0; \tag{3}$$

$$\ell_{pq} < g_{pq} < b_{pq} \quad \text{all OD pair } pq \tag{4}$$

$$f_a = \sum_p f_a^p \ ; \tag{5}$$

where

W_i = set of links beginning at i
V_i = set of links ending at i
δ_{pi}= 1 if i is a destination node; 0 otherwise
ℓ_{pq}= a lower bound of g_{pq}
b_{pq}= an upper bound of g_{pq}.

The demand g_{pq} is considered here to represent the flow on a fictitious return link from destination q to origin p. The travel cost on a fictitious return link (q,p) is equal to $-G_{pq}(g_{pq})$. ℓ_{pq} may be set equal to any small positive value as $G_{pq}(0)$ is generally undefined. If no natural upperbound of g_{pq} is available then b_{pq} may be any sufficiently large positive value to ensure a bounded feasible region. In addition, ℓ_{pq} and b_{pq} may also be used to model the captive effect of a given transportation mode whenever the demand function is a modal split function.

Since feasible direction methods in nonlinear programming were applied successfully to the traffic assignment problem with fixed demand, we will follow a similar approach and proceed to adapt the Frank and Wolfe (FW) method [16] and the convex-simplex (CS) method [17] to solve the above problem. This adaptation consists essentially in the simplification of the computational procedure of the FW and CS algorithms by exploiting the special structure of the traffic assignment problem.

III The development of the algorithms

1. An adaptation of the Frank and Wolfe method.

The traffic assignment problem P may be written in the following standard form:

$$\text{minimise} \quad F(x) \qquad (6)$$
$$\text{subject to } Ax = 0 \ , \ x \geq 0,$$

where F is convex and continuously differentiable. Let x^1 be a feasible solution to (6). The Frank and Wolfe method determines the next improved solution x^2 as follows: first it determines a direction $d = y - x^1$, where y is an optimal solution to the following LP subproblem:

$$\text{(LP)} \qquad \text{minimise} \quad \nabla F(x^1)\, y$$
$$\text{subject to } Ay = 0, \ y \geq 0,$$

where $\nabla F(X^1)$ is the transpose of the gradient of F at x^1, and then it determines the optimal step lenght λ^* to be made in that direction by minimizing $F(x^1 + \lambda d)$ for $0 \leq \lambda \leq 1$. The solution x^2 is set equal to $x^1 + \lambda^*(y-x^1)$. Should x^1 be an optimal solution one obtains $\nabla F(x^1)\ (y-x^1) = 0$.

Let $\{f_a^p, \bar{g}_{pq}\}$ be a feasible solution of P. The above LP subproblem is written as:

$$\text{minimise} \quad \sum_p \{ \sum_a \bar{c}_a \ y_a^p - \sum_g \bar{u}_{pq} \ z_{pq} \} \qquad (7)$$

subject to

$$\sum_{a \in W_i} y_a^p - \sum_{a \in V_i} y_a^p + \delta_{pi} \ z_{pi} = 0, \ i \neq p,$$
$$i = 1, \dots, n \ ; \ p = 1, \dots, n_0$$
$$y_a^p \geq 0 \qquad \text{all a, all p}$$
$$\ell_{pq} \leq g_{pq} \leq b_{pq} \qquad \text{all pq}$$

$$\text{where } \bar{c}_a = C_a(\bar{f}) \text{ and } \bar{u}_{pq} = G_{pq}(\bar{g}_{pq})$$

Since there is no binding constraint on the arc flows y_a^p associated with different origins p, the LP subproblem (7) may be solved separately for each origin. For a given origin p, the associated subproblem is an "all or nothing" traffic assignment with variable demand, where the travel cost on link a is $\bar{c}_a$ and the desired interzonal travel cost between origin p and destination q is $\bar{u}_{pq}$. Solving this subproblem is rather trivial, an optimal solution may be obtained as follows: let v_{pq} denote the travel cost on the shortest path from origin p to destination q. If v_{pq} is different from $\bar{u}_{pq}$ then the trial demand z_{pq} is either set equal to its lower bound ℓ_{pq} or to its upper bound b_{pq}; otherwise z_{pq} is held at the current value $\bar{g}_{pq}$. $\{y_a^p\}$ is the resulting all or nothing traffic pattern obtained by loading the demand z_{pq} onto the shortest paths between p and q. More precisely

$$z_{pq} = \begin{cases} \ell_{pq} & \text{if } v_{pq} > \bar{u}_{pq}, \\ \bar{g}_{pq} & \text{if } v_{pq} = \bar{u}_{pq}, \\ b_{pq} & \text{otherwise;} \end{cases}$$

and $$y_a^p = \sum_q \delta_{aq} z_{pq}$$

where δ_{aq} equals 1 if link a belongs to the shortest path between p and q, and 0 otherwise,

The direction d is therefore given as follows:

$$d_{pq} = z_{pq} - \bar{g}_{pq}$$

$$d_a^p = y_a^p - \bar{f}_a^p$$

$$d_a = \sum_p d_a^p = \sum_p \sum_q \delta_{aq} z_{pq} - \bar{f}_a$$

The above direction leads to the following new traffic pattern:

$$g_{pq} = \bar{g}_{pq} + \lambda^* d_{pq}$$

$$f_a^p = \bar{f}_a^p + \lambda^* d_a^p$$

$$f_a = \bar{f}_a + \lambda^* d_a$$

where λ^* solves:

$$\min Z(\bar{f}_a + \lambda d_a, \bar{g}_{pq} + \lambda d_{pq}) ; \quad 0 < \lambda < 1.$$

Note that the explicit knowledge of the flows f_a^p associated with each origin p is not necessary in the above procedure.

The physical interpretation of the above procedure is simple. Recall that $\bar{u}_{pq}$ is the interzonal travel cost that would induce a demand equal the current demand according to the demand function. If $\bar{u}_{pq}$ is less than the travel cost on the current shortest path between p and q, the flow on all utilized paths between p and q will be decreased. On the contrary, if $\bar{u}_{pq}$ is greater than the travel cost on the shortest path then the demand g_{pq} will be increased: this additional quantity will be added to the shortest path simultaneously with a transfer of flow from all utilized paths to the shortest path. Finally, if $\bar{u}_{pq}$ is not different from the travel cost on the shortest path then only a transfer of flow from all utilised paths to the shortest path will occur.

Based on the convexity of the objective function Z, it is possible to compute the following lower bound of Z at each iteration:

$$LB = Z(\bar{f}_a, \bar{g}_{pq}) + \nabla Z(\bar{f}_a, \bar{g}_{pq}) \times (d_a, d_{pq})$$

where ∇Z denotes the transpose of the gradient of Z. This lower bound may be used as a stopping criterion in the following resulting computational procedure:

Algorithm E_1

Given an initial traffic pattern $\{f_a, g_{pq}\}$

Step 1 Compute the current travel cost $c_a = C_a(f_a)$ and $u_{pq} = G_{pq}(g_{pq})$.

Step 2 For each OD pair pq determine the travel cost v_{pq} on the current shortest path. Set $\alpha = (v_{pq} - u_{pq}) / v_{pq}$.

If $\alpha > \varepsilon$ (a suitable tolerance) set $z_{pq} = \ell_{pq}$ (lower bound on the travel demand).

If $|\alpha| < \varepsilon$, set $z_{pq} = g_{pq}$; otherwise set $z_{pq} = b_{pq}$ (upper bound on the travel demand.

Load z_{pq} onto the current shortest path from origin p to destination q to obtain all or nothing traffic pattern: $\{y_a, z_{pq}\}$.

Step 3 If the current traffic pattern $\{f_a, g_{pq}\}$ satisfies a user's stopping criterion, terminate.

Step 4 Determine the step length λ^* by solving

$$\min Z\,(f_a + \lambda(y_a - f_a),\ g_{pq} + \lambda(z_{pq} - g_{pq}))$$

subject to $0 < \lambda < 1$.

Step 5 Revise the flows and demands as

$$f_a = f_a + \lambda^*\,(y_a - f_a)$$

$$g_{pq} = g_{pq} + \lambda^*\,(z_{pq} - g_{pq})$$

Return to step 1.